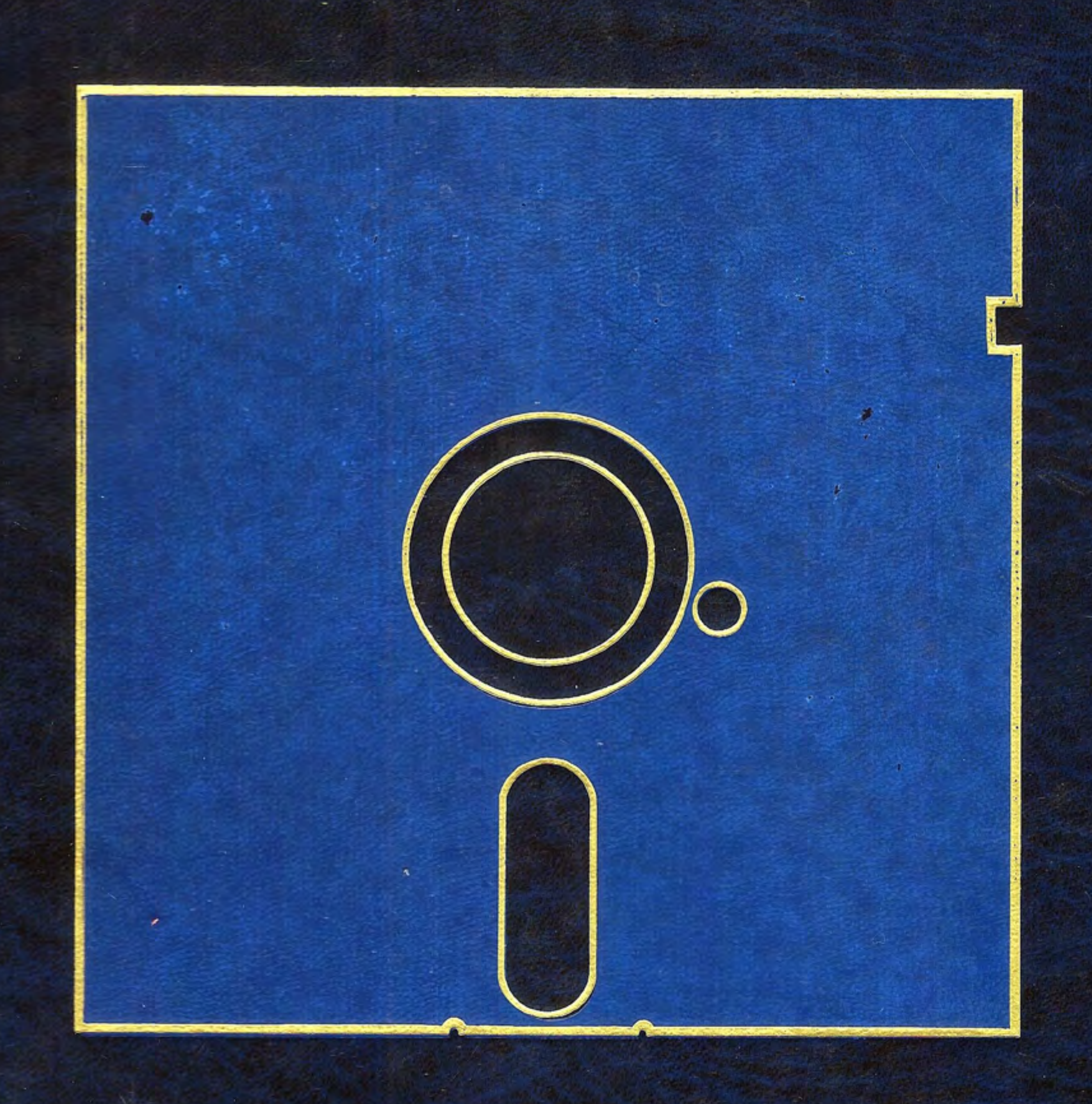
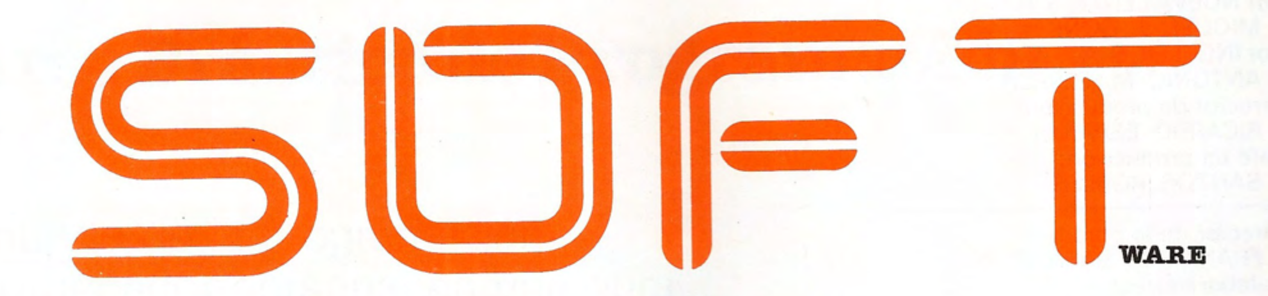
SOFTWARE





Nueva Lente/Ingelek

una publicación

EDICIONES NUEVA LENTE, S. A.,
y EDICIONES INGELEK, S. A.

Director-Editor:
Por NUEVA LENTE, S. A.:
MIGUEL J. GOÑI
Por INGELEK, S. A.:
ANTONIO M. FERRER
Director de producción:
RICARDO ESPAÑOL
Jefe de producción:
SANTOS ROBLES

Director de la obra:
FRANCISCO LARA (PL/3)
Colaboradores:
MANUEL MUÑOZ
DAVID SANTAOLALLA
SANTIAGO RUIZ
LUIS COCA
MIGUEL ANGEL VILA
MIGUEL ANGEL SANCHEZ
VICENTE ROBLES

Diseño e ilustración:
JOSE OCHOA (PL/3)
Maquetación:
JUAN JOSE DIAZ SANCHEZ
Fotografía:
(Equipo Gálata)
EDUARDO AGUDELO y
ALBINO LOPEZ
Redacción:
PL/3 calc

Dirección, Redacción y
Administración:
Benito Castro, 12. 28028 Madrid
Tels. 245 45 98 y 246 73 67

ISBN, tomo primero: 84-7534-103-9
ISBN, tomo segundo: 84-7534-104-7
ISBN, tomo tercero: 84-7534-105-5
ISBN, tomo cuarto: 84-7534-106-3
ISBN de la obra: 84-7534-101-2

Fotomecánica:
OCHOA, S. A.
Ricardo Ortiz, 74. 28017 Madrid
Impresión:
GRAFICAS REUNIDAS, S. A.
Avda. de Aragón, 56. 28027 Madrid
Depósito legal: M. 41.955-1984
PRINTED IN SPAIN



Curso Práctico de Programación

El mundo de la programación del ordenador personal en una obra práctica, completa, didáctica y actualizada

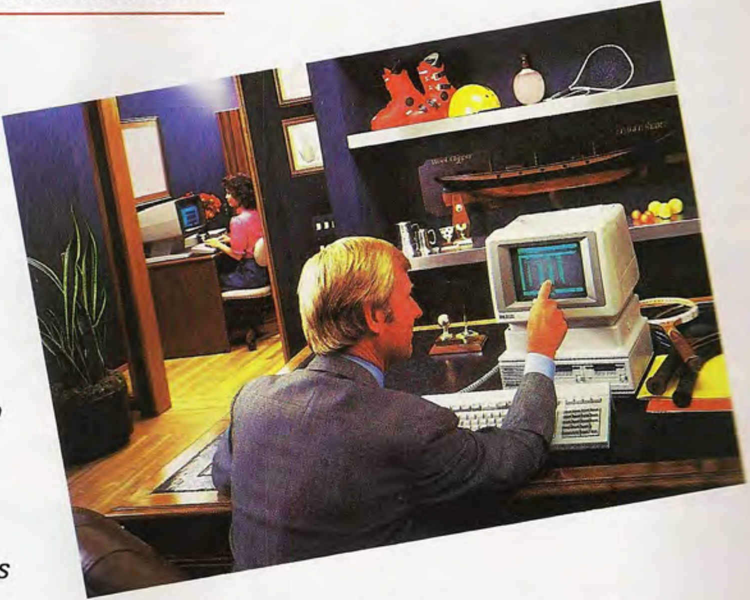
MUCHOS hogares cuentan ya con un nuevo huésped. Un invitado que llena los ratos de ocio llevando la espectacularidad de los juegos de acción a la pantalla del televisor, colaborando en la gestión de la contabilidad doméstica, controlando los movimientos de las cuentas bancarias, ayudando al estudiante de la casa, o sugiriendo el menú adecuado para la próxima semana.

Cada día son más y más las oficinas que han sustituido los pesados libros contables, las máquinas de escribir, e incluso las tradicionales calculadoras, por una nueva herramienta, versátil, inteligente y eficaz.

No cabe duda, el protagonista de la década es, con todos los honores, el ordenador personal.

En muy pocos años, estas útiles máquinas han saltado de los libros de ciencia ficción y de las fantasías del celuloide a la cotidianidad.

Mucho se ha avanzado en su conocimiento. Palabras como bit, microprocesador, hardware o software han perdido una gran parte de su misterio e impenetrabilidad para muchas personas. Pero siguen quedando muchas preguntas sin respuesta para los que se han incorporado más recientemente al mundo de la informática; o preguntas con una respuesta sólo apuntada, no definitiva, clara y detallada, para los más versados.



¿Qué es y para qué sirve un ordenador personal?

¿Qué soluciones aporta un ordenador personal en el hogar, en el despacho de un abogado, en la consulta de un médico, en el estudio de un escritor, en un comercio, en la empresa..?

¿Qué datos hay que sopesar para adquirir el ordenador personal adecuado para cada situación y necesidad?

¿Cómo obtener el máximo rendimiento del ordenador personal, conociendo sus posibilidades y limitaciones?

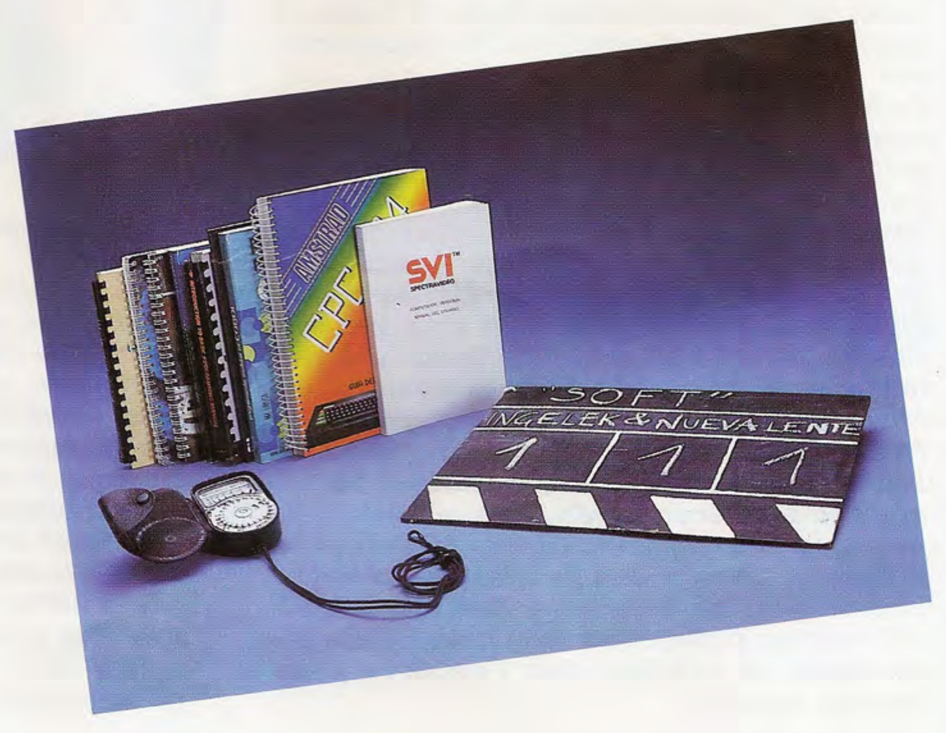
¿Sabe cómo «instruir» al ordenador personal para que actúe de acuerdo a sus necesidades específicas?

¿Qué tareas puede programar y cómo hacerlo?





SOFTWARE nace para dar una respuesta práctica a las cuestiones planteadas y a otros muchos interrogantes que convergen en el protagonista de la nueva era microinformática.
El ordenador personal es una máquina dispuesta a prestar una colaboración inapreciable, capaz de resolver una ingente variedad de aplicaciones.
En su esencia más íntima, el ordenador personal no difiere mucho de otros dispositivos e instrumentos electrónicos destinados a una función específica. La transformación del conglomerado de circuitos electrónicos en máquina inteligente, en ordenador, obedece a un concepto mágico: la *programación.*
Para convertir a la máquina programable en un útil colaborador, es preciso conocerla, saber cómo rodearla de los periféricos adecuados para cada necesidad y, desde luego, aprender a instruirla para que realice con eficacia y premura las tareas más dispares.
En definitiva, la actuación del ordenador depende por completo de la *programación* que reciba. Por lo tanto, hay que aprender a dialogar con la máquina: dominar su lenguaje, conocer los sistemas operativos que constituyen su inteligencia elemental y rigen su funcionamiento, y familiarizarse con los programas y paquetes de aplicación estandarizados que convierten al ordenador personal en un eficiente colaborador.
Tres son los eslabones que dan cuerpo al concepto de programación: los *sistemas operativos,* los *lenguajes de programación* y el *software de aplicación.* Su coexistencia en la máquina convierte a ésta en un ordenador, capaz de acometer y llevar a buen término las más diversas tareas.



¿QUE ES Y PARA QUE SIRVE EL ORDENADOR PERSONAL?

En su definición más simple, el ordenador personal no es más que una máquina, de reducidas dimensiones y precio moderado, cuyo cerebro está regido por un circuito integrado programable: el microprocesador.

Tal definición es extensiva a la generalidad de equipos destinados al tratamiento de información, cuya unidad central de proceso está organizada en torno a un microprocesador: los microordenadores. Dentro de esta gran familia, caben potentes equipos para la gestión de tareas complejas y sofisticadas, y microordenadores capacitados para distribuir su atención entre varios usuarios simultáneamente. Y, por supuesto, toda la enorme variedad de ordenadores personales.

La distinción entre el ordenador personal y los restantes microordenadores, cabe precisarla en su objetivo: equipos destinados al tratamiento de información gobernados por un usuario individual. Una característica que afecta a su ámbito de explotación, individualizado, no multiusuario. Hay ordenadores personales especialistas en la vertiente más lúdica (juegos de acción, de reflexión, de estrategia...); inclinados a la enseñanza asistida por ordenador; versados en la confección del correo personal y el control de las cuentas domésticas; verdaderos expertos en la gestión de archivos y en el tratamiento de datos; virtuosos en el arte de automatizar el hogar; consumados expertos en la planificación de supuestos financieros; eficaces y potentes gestores de las tareas de administración en el ámbito de una pequeña o mediana empresa...





Una obra completa y actualizada, organizada en cuatro secciones que recogen todo el mundo de la programación del ordenador personal.



Basic

Curso práctico de lenguaje BASIC, particularizado para los ordenadores personales más populares. Con un tratamiento gráfico, didáctico y repleto de programas y ejemplos.

Lenguajes

Un estudio completo y detallado de los principales lenguajes para ordenadores personales.

Sistemas Operativos

La inteligencia elemental del ordenador al descubierto, con un pormenorizado análisis práctico de los sistemas operativos que predominan en el campo de los ordenadores personales.

Aplicaciones

Una guía práctica para elegir y aprender a utilizar los programas y paquetes de aplicación que convierten al ordenador personal en un colaborador eficiente y versátil.

Todo ello completado con una amplia variedad de cuadros, complementos e ilustraciones que familiarizarán al lector con los conceptos básicos de la informática del ordenador personal.



El ordenador personal. Todo un símbolo de la sociedad moderna nacido de la feliz idea que asaltó a Jonathan Titus, en 1974: utilizar un circuito integrado de la firma americana Intel —cuya referencia era Intel 8008 y al que denominaban microprocesador—, para crear un pequeño ordenador destinado a los aficionados al bricolage electrónico... Ocho años bastaron para que en 1982 la prestigiosa revista TIME nombrara «Personaje del año» al ordenador personal.



Curso Práctico de Programación

CURSO DE BASIC

Si hubiera que destacar un lenguaje dentro del universo de los ordenadores personales, éste sería, sin lugar a dudas, el BASIC. Un lenguaje popular y compartido por casi la totalidad de los equipos presentes en el mercado.

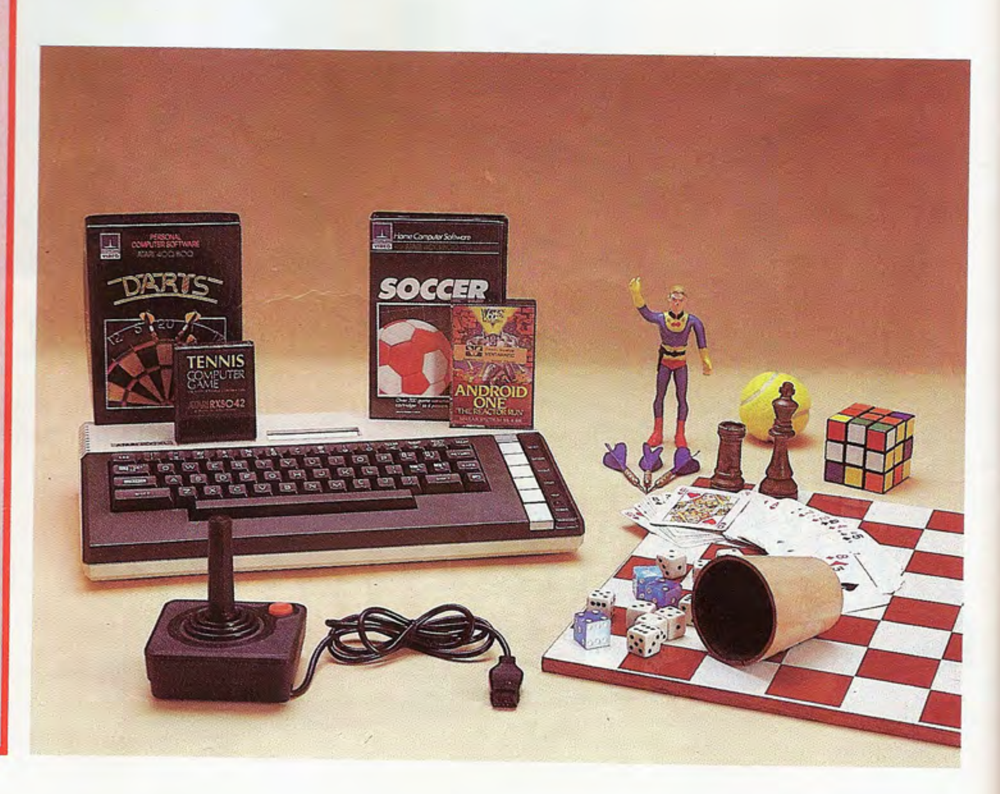
El desarrollo del CURSO DE BASIC incluye un estudio completo y detallado, repleto de ejemplos, del vocabulario y la sintaxis del lenguaje BASIC; dedicando una constante atención a los dialectos que constituyen el idioma de los ordenadores personales de mayor difusión y popularidad.

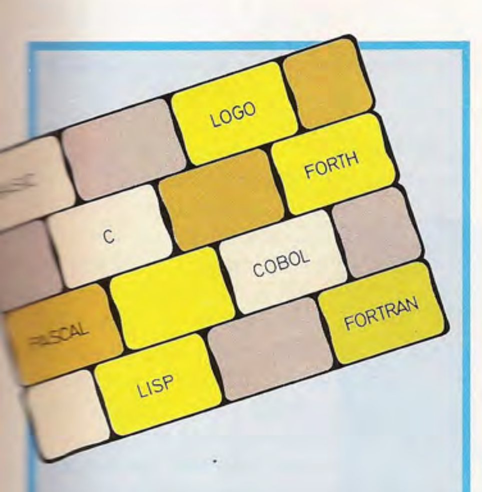
Paralelamente, se describen las técnicas y métodos de programación que permitirán al lector confeccionar sus propios programas, cada vez más complejos y potentes, adecuados para resolver múltiples aplicaciones.

A lo largo del curso, el lector encontrará un dilatado abanico de programas de juegos, educativos, de gestión y utilidades que guiarán su aprendizaje a través de la práctica.

Los dialectos considerados dentro del CURSO DE BASIC, corresponden a ordenadores tan populares como ZX-SPECTRUM, ZX-81, COMMODORE-64, VIC-20, DRAGON, ATARI, SPECTRAVIDEO, ORIC, NEW BRAIN, SHARP, AMSTRAD, Equipos MSX, IBM-PC, APPLE, DIGITAL, OLIVETTI, TOSHIBA, NCR...







LENGUAJES

A la hora de establecer un diálogo con el ordenador, el usuario no sólo cuenta con el lenguaje BASIC. Hay otros lenguajes informáticos, asentados en el mundo del ordenador personal, que coexisten con el BASIC y que incluso superan a éste último en determinadas áreas: enseñanza, juegos, tareas científicas, de gestión...

Además de aportar los conocimientos básicos relativos al mundo de los lenguajes, esta sección abordará el estudio práctico de los lenguajes más importantes en el ámbito del ordenador personal. Entre ellos cabe destacar los siguientes:

- LOGO
- PASCAL
- FORTRAN
- COBOL
- FORTH
- «C»
- ADA
- LISP
- APL
- PILOT

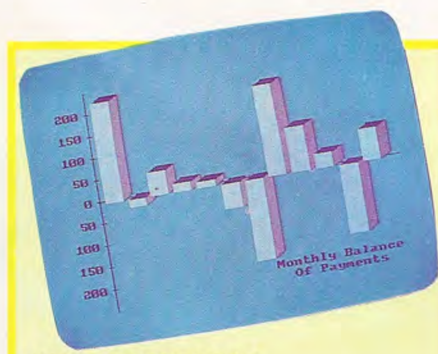
SISTEMAS OPERATIVOS

Los cimientos del edificio de la programación los aporta la circuitería, el hardware del ordenador personal. Sobre esta base, hay que empezar colocando la estructura que acogerá a los traductores de lenguajes y a los programas y paquetes de aplicación.

La estructura que soportará a los restantes elementos de la programación no es más que el sistema operativo. Una inteligencia elemental que da pie al desarrollo del ordenador, y que condiciona su expansión en el terreno de la programación. Tanto los traductores de lenguajes como los paquetes de aplicación entran al sistema a través de la vía de compatibilidad que abre el sistema operativo.

Son muchos los sistemas operativos concebidos para regir la actividad del ordenador personal. Entre ellos cabe destacar un grupo que ostenta el liderazgo casi absoluto y que reciben, dentro de esta sección, un tratamiento detallado y didáctico:

- CP/M y toda su familia de derivados (CP/M-80, CP/M-86, MP/M...)
- MS/DOS (PC/DOS, MSX/DOS, XENIX)
- APPLE DOS
- PRO.DOS
- OASIS
- PICK
- UNIX
- UCSD
- FLEX



APLICACIONES

Nada más frecuente que la imagen del ordenador resolviendo un complicado análisis financiero, confeccionando la nómina de la empresa, colaborando en el proceso de textos, gestionando una completa y detallada base de datos, o aportando su inteligencia al tiempo de ocio con emocionantes juegos.

A la hora de proveer al ordenador de los programas adecuados para cada aplicación, el usuario dispone de varias alternativas: confeccionar sus propios programas, encargar su confección «a medida», o adquirir programas estandarizados que puedan resolver sus necesidades específicas.

Los grupos más importantes de programas y paquetes de aplicación tienen un lugar en SOFTWARE. Un estudio completo y detallado que se completa con el análisis práctico de los principales paquetes destinados a aplicaciones de gestión:

- Tratamientos de textos.
- Hojas electrónicas.
- Sistemas para la gestión de bases de datos.
- Software gráfico.
- Generadores de programas.
- Paquetes para comunicaciones.
- Gestionadores de tiempo y tareas.
- Paquetes integrados.

El ordenador personal

El símbolo de la nueva sociedad informática

A lo largo de su historia, la humanidad ha asistido a grandes revoluciones. Revoluciones desencadenadas, en muchos casos, por innovaciones tecnológicas. Algunas llegaron a influir tan drásticamente en el desarrollo de la sociedad, que abrieron una nueva era en la historia.

Uno de los principales condicionantes de la evolución humana son los métodos de trabajo de sus protagonistas; y estos métodos se ven alterados por ciertos destellos en el ámbito de la tecnología. No hay más que pensar en el efecto que tuvieron, en sus respectivas sociedades, la invención del fuego, la rueda o el arte de trabajar el hierro.

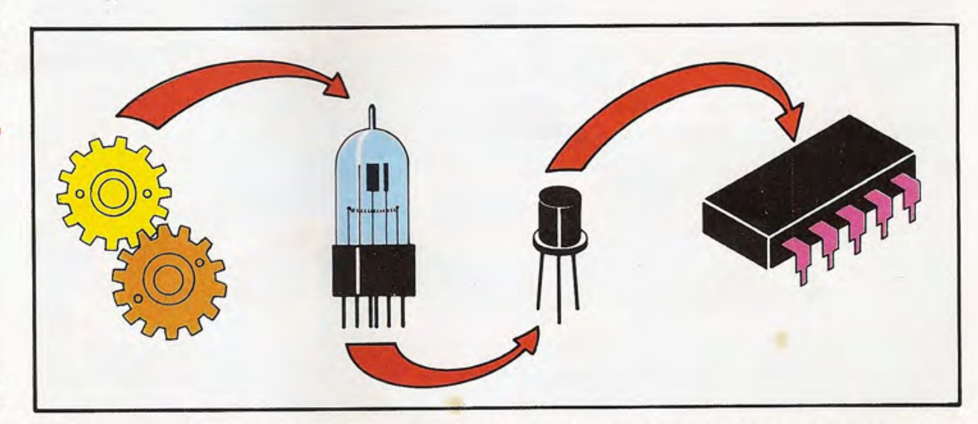
En el siglo XVIII se encuentra el penúltimo gran eslabón de la cadena de revoluciones tecnológicas: la revolución industrial. La producción artesanal había agotado todas sus posibilidades; los productos escaseaban y alcanzaban precios desorbitados. Era necesario crear máquinas capaces de aumentar las posibilidades de producción. Los nuevos métodos de producción convulsionaron la economía mundial. Los países que no supieron o no pudieron estar al día, quedaron apeados durante mucho tiempo, y en algunos casos de forma casi irreversible, del camino hacia la sociedad actual.



Con la entrada en escena del microprocesador, los ordenadores han reducido su tamaño y precio. En la actualidad su presencia es habitual en cualquier ambiente: oficinas, despachos profesionales, e incluso en el hogar.

LA REVOLUCION INFORMATICA

A finales del siglo XIX y principios del XX, empezó a observarse un estancamiento en el terreno de la administración. Empresas de envergadura encontraban dificultades para tratar y archivar el flujo de infor-



Los primeros ordenadores fueron mecánicos. Más tarde, en un primer salto tecnológico, pasaron a estar constituidos por válvulas, luego por transistores y, en la actualidad, por circuitos integrados.

mación que las asaltaba. Paralelamente, algunas disciplinas del saber científico empezaron a quedar vedadas a un mayor conocimiento, dada la imposibilidad de acometer los voluminosos cálculos que exigían.
La mente humana empezó a buscar soluciones: las máquinas no sólo debían manufacturar objetos, sino también información.
Ya en el siglo XVIIIse habían esbozado algunas ideas tendentes al tratamiento automático de la información. Se pensó en la posibilidad de crear artilugios mecánicos capaces de realizar este cometido. Aunque era posible concebirlos en el plano teórico, pronto se comprobó que existían un puñado de buenas razones que hacían inviable el desarrollo de estas máquinas.
Estrenado el siglo XX, surgió la idea de utilizar dispositivos electromecánicos. Estos no dieron mal resultado; sin embargo, tenían muchas limitaciones. Era preciso dar un nuevo salto. Este se consumó al descubrir los primeros dispositivos electrónicos, que relegaron al olvido la idea del ordenador mecánico.
En una fecha reciente, a mediados de este siglo, los descubrimientos en el campo de la electrónica permitieron el nacimiento de los primeros ordenadores modernos. Empezaron a construirse a partir de válvulas de vacío. Más tarde con transistores. Y, por último, a base de circuitos integrados. Los primeros ordenadores eran voluminosos cajones llenos de parafernalia electrónica, destinados, principalmente, a aplicaciones científicas y militares. Más tarde, fueron reduciendo su tamaño y precio. Hasta llegar a la actualidad: con la irrupción del microprocesador (su cerebro integrado), los ordenadores han sorteado totalmente la barrera de volumen y precio, llegando a ser asequibles al gran público.
Hoy en día, un ordenador está al alcance de cualquier bolsillo; ha franqueado el umbral del hogar. Y tal vez, muy pronto, se convierta en un electrodoméstico tan imprescindible como puede serlo el televisor o la lavadora automática.

¿QUE ES UN ORDENADOR?

Un ordenador es una máquina programable cuyo cometido es el tratamiento automático de la información. Una actividad genérica que se sintetiza en tres fases: *entrada de la información, tratamiento* o procesado de la misma, y *salida* de la información resultante del tratamiento.
Para acometer esta tarea genérica, el ordenador cuenta en su interior con dispositivos electrónicos capaces de transformar la información en señales eléctricas *(entrada)*. Su propia circuitería electrónica manipulará las señales eléctricas de acuerdo a las instrucciones recibidas *(tratamiento de la información)*. Y, por último, las señales eléctricas resultantes serán convertidas, por obra y gracia de los dispositivos electrónicos, en una información inteligible *(salida)*.
La celeridad con la que actúan los dispositivos electrónicos, permiten al ordenador realizar múltiples operaciones en unas pocas fracciones de segundo. Ello convierte al ordenador en una herramienta rápida y eficiente en un sinfín de actividades relacionadas, todas ellas, con el tratamiento de la información. Por ejemplo, puede clasificar y archivar grandes lotes de información, con rapidez y fiabilidad. Es capaz de trabajar con datos numéricos, operando densos y complicados cálculos. Y también resulta eficaz en las actividades de control de procesos: puede tratar la información procedente de un conjunto de sensores y controlar a otros instrumentos asociados al ordenador.
Algunos representantes de esta gran familia de máquinas programables, los ordenadores personales, han saltado ya a los escaparates de las tiendas de electrodomésticos.



La revolución microinformática ha puesto al ordenador personal al alcance del gran público. Tal vez se convierta, muy pronto, en un útil tan imprescindible como una calculadora o una máquina de escribir.

TOMA DE CONTACTO CON EL ORDENADOR

Mucho es lo que queda por añadir acerca de la naturaleza y posibilidades del ordenador; todo un compendio de conocimientos que se irán exponiendo e ilustrando a lo largo de esta obra. Por el momento, es suficiente con una somera idea inicial. Quien debe presentarse ahora es el propio ordenador, tal y como se nos revela al abrir la sugerente caja que nos entregan en la tienda.
La adquisición de un pequeño ordenador personal suele ser, para muchos, el primer paso en el sendero de la informática; y la solemne apertura de la caja, su primer contacto con la realidad del ordenador.

Una vez desempaquetado, nos encontramos con un pequeño y liviano mueble de plástico, repleto de teclas y tomas de conexión: un ordenador, ni más ni menos. En el fondo de la caja tropezamos con su corte de acompañantes: una pequeña caja, pesada y compacta, de la que parten dos cables, un manual y otros dos mazos de cable con sus respectivos conectores en cada extremo. En definitiva, todo lo necesario para que nuestro flamante ordenador pueda demostrar sus habilidades.

La pesada caja flanqueada por sendos cables, no es más que el alimentador, a través del cual el ordenador recibirá la energía tomada de la tensión de la red eléctrica. El extremo de uno de los cables se conectará a la toma que el ordenador posee al efecto, mientras que el otro extremo se insertará en un enchufe de la red eléctrica.

Los dos cables que viajan separados son, respectivamente, para la conexión de una «pantalla» (el órgano de visualización) y para la adaptación al ordenador de un dispositivo de memoria externa. En la mayor parte de los ordenadores, la pantalla de visualización puede ser un simple televisor doméstico. Uno de los extremos del cable al efecto se conectará a la entrada de antena del receptor de TV y el otro al ordenador.

Con el ordenador conectado a la tensión de red, a través del alimentador, y asociado al receptor de TV, ya se puede empezar a trabajar. No obstante, si se desea conjuntar un sistema realmente operativo, hacen falta aún otros complementos.

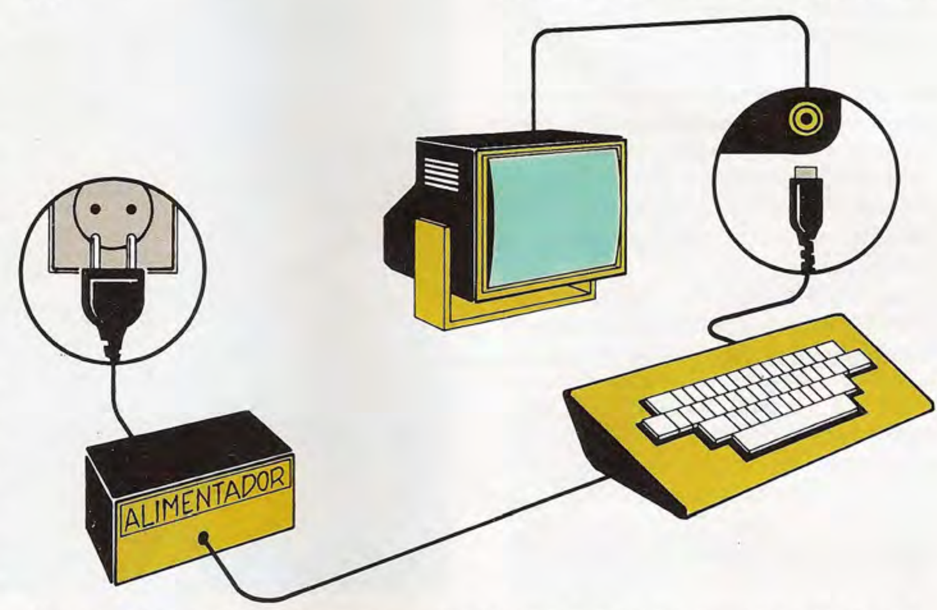
Para almacenar la información en un soporte permanente y no tener que reescribirla cada vez que se enciende el ordenador, es preciso contar con un *periférico* de almacenamiento: un dispositivo capaz de grabar, memorizar y reproducir información. Este puede ser un simple magnetófono a casetes o una unidad para discos flexibles. En un ordenador doméstico, la alternativa inicial suele coincidir con el magnetófono a casetes. Y ahí está el cometido del cable que permanecía en la caja junto con el manual; un extremo se conectará al magnetófono y el otro al ordenador.

El otro componente básico de un sistema ordenador será un periférico que nos permita obtener una copia de la información, de tal forma que podamos verla sin necesidad de mantener encendido el aparato. En efecto, nos referimos a la impresora: un dispositivo externo capaz de obtener copias en papel de la información contenida en el ordenador.

Al observar la zona posterior y los laterales del mueble del ordenador, se observa que aún quedan algunos conectores sin utilizar. Un repaso al manual nos permitirá deducir cuál es el cometido específico de cada uno de estos *conectores de expansión*.

EL ORDENADOR Y SU ENTORNO

La toma de contacto con el ordenador nos ha permitido esbozar una idea muy impor-



¡Vamos a dialogar en BASIC con el ordenador! Para ello, habrá que empezar rodeándolo de los complementos imprescindibles: un alimentador, a través del que recibirá la energía necesaria, y una pantalla (monitor de vídeo o receptor de TV) en la que plasmar los mensajes.

Una breve historia del BASIC

El lenguaje de programación BASIC (Beginner's All-Purpose Symbolic Instruction Code) nació en 1964 en el Dartmouth College, de la mano de John G. Kemeny y Thomas Kurtz, y fue concebido como un lenguaje interactivo, polivalente y de fácil aprendizaje y empleo.

En un principio fue normalizado por el organismo ANSI (American National Standards Institute) y de esta normalización parten las líneas originales del BASIC. Más tarde, surgió toda una gran familia de dialectos que cada vez se fueron desviando más y más del lenguaje original.

En 1977, la empresa americana Microsoft desarrolló un dialecto que pretendía unificar criterios. Rápidamente fue aceptado por varios fabricantes de ordenadores como Tandy, Apple, Commodore...

El gran boom del BASIC ha llegado con la irrupción de los microordenadores, con la gran ventaja de su precio, que los ha hecho asequibles a cualquier bolsillo. Pero hay que señalar que en un principio el BASIC fue adoptado por los sistemas comerciales de tiempo compartido. De éstos es de donde viene la popularidad del BASIC.

En la década de los ochenta, el BASIC se ha constituido en el lenguaje de programación más utilizado. Aunque se habla de varios lenguajes como futuros sustitutos del BASIC, lo cierto es que ninguno amenaza seriamente la posición privilegiada que éste mantiene en el campo del ordenador personal.

La adquisición de un pequeño ordenador personal constituye, para muchas personas, su primer paso en el sendero de la informática. Un primer paso que conlleva la exigencia de familiarizarse con el lenguaje del ordenador: el BASIC sin lugar a dudas.



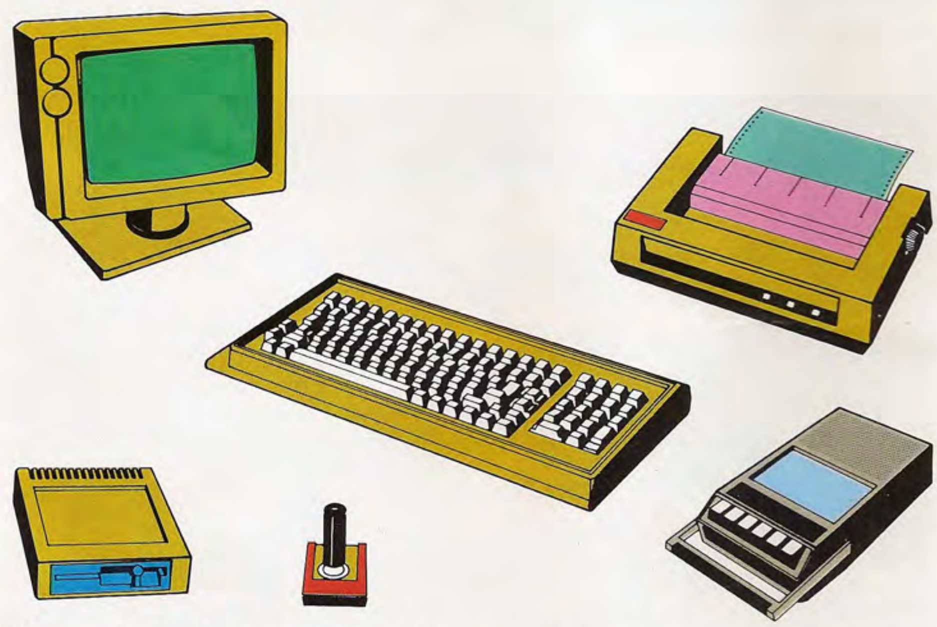
tante. Para que éste se convierta en un instrumento plenamente utilizable, es preciso rodearlo de algunos periféricos o dispositivos externos: hay que construir un sistema ordenador. Como ya hemos observado, los dispositivos periféricos son los que permiten al ordenador establecer una comunicación con el usuario o universo exterior a la máquina. Estos pueden dividirse en tres grandes bloques: periféricos de entrada, periféricos de salida y unidades para el almacenamiento de información.

El periférico de entrada más común es el teclado; su importancia es tal que, normalmente, suele suministrarse junto con la unidad central del ordenador. Otros periféricos de entrada, frecuentes en los ordenadores domésticos, son los mandos para juegos: joysticks (palancas de juego), paddles, lápices ópticos, tabletas gráficas...

Los periféricos de salida son los que trasladan al exterior la información de salida o respuestas del ordenador. Dentro de este grupo se encuentran las pantallas (ya sea un monitor de vídeo o un simple receptor de TV), las impresoras...

Por último, hay que hablar de las unidades para almacenamiento de información. Estos son periféricos que permiten al ordenador conservar la información que pueda necesitar en cualquier otro momento. Su actuación se plasma en dos operaciones: escritura (grabación) y lectura (recuperación) de la información procedente o destinada al ordenador. Dentro de esta categoría destacan las unidades de casete y las unidades de disco flexible; aunque también cabe hablar de las unidades de disco rígido, de cinta magnética, etc.

Además de los dispositivos periféricos, cabe mencionar a otros complementos o medios de expansión que tienen por objeto potenciar las posibilidades del sistema ordenador. Estos *útiles de expansión* pueden ser circuitos electrónicos que incrementan la potencia de cálculo, amplían la zona de memoria interna del ordenador, o permiten la conexión al sistema de otros tipos de periféricos. Estos últimos medios de expansión, de adaptación



Para que el ordenador se convierta en un instrumento con plena eficacia práctica, es preciso rodearlo de algunos periféricos o dispositivos externos. Los acompañantes más frecuentes del ordenador personal son: el teclado, para comunicarle las órdenes y mensajes; la pantalla u órgano de visualización; las unidades para el almacenamiento de información (cinta de tipo casete o discos flexibles); la impresora, para obtener copias en papel de la información procesada, y los periféricos para juegos, entre los que destaca el popular «joystick».

o interface, compatibilizan el formato de comunicación del ordenador con los dispositivos periféricos.

ESTRENANDO EL DIALOGO

El ordenador está ya en disposición de entrar en funcionamiento. Dada su actividad —entrada, tratamiento y salida de información—, es obvio que la atención del usuario debe empezar por centrarse en las dos «vías» a través de las que se establecerá el diálogo. El camino de entrada, para «hablar» con el ordenador, lo aporta el teclado; un medio que resultará familiar ya que, salvo la presencia de algunas teclas especiales, su semejanza es total con el de una máquina de escribir. Así, pues, para entablar una comunicación con el ordenador, habrá que «teclear» las palabras. La vía de salida de información no exige mayores detalles puesto que se trata de una simple pantalla. En ella aparecerán las «respuestas» de la máquina.

El alimentador está ya enchufado a la tensión de red; basta sólo con accionar el interruptor de encendido para dar vida a la máquina. Tras un ligero parpadeo de la pantalla, ésta visualiza un mensaje de presentación, distinto según el modelo de ordenador. Bajo éste, aparece un elemento que a partir de ahora se convertirá en familiar: el *cursor*.

El cursor es el nombre que recibe el sím-

El teclado

El periférico o dispositivo más frecuente a través del que hablaremos con la máquina es el teclado. Tal como ocurre con las máquinas de escribir, éste suele presentar ligeras diferencias según el modelo, si bien su estructura es muy parecida.

El teclado de un ordenador incorpora una zona básica, cuyas teclas son muy semejantes a las de una máquina de escribir; esta zona es la que se denomina «teclado alfanumérico». En ella están agrupadas las letras del alfabeto, las cifras decimales y algunos símbolos y caracteres ortográficos.

La diferencia respecto a la máquina de escribir se debe a la presencia de algunas teclas especiales, útiles para comunicar al ordenador ciertas órdenes que condicionarán su actividad.

Algunos teclados de ordenadores pueden omitir la presencia de ciertas teclas especiales, incorporar otras, o alterar su nombre. No obstante, las más frecuentes son las que se relacionan en el gráfico adjunto.

RETURN (ENTER en otros teclados): hay que accionarla al terminar la introducción de una orden o mensaje. Al pulsarla, el ordenador «asimilará» el mensaje introducido.

CONTROL: cambia la función de algunas teclas con objeto de introducir una orden o mensaje de control.

INSERT (Insertar): se utiliza para insertar nuevos caracteres en medio del texto anteriormente escrito.

DELETE (Borrar): hace retroceder el cursor una posición, borrando el carácter que la ocupaba.

BREAK (Ruptura): rompe con la tarea que está ejecutando el ordenador.

RESET (Inicialización): devuelve al ordenador a su estado inicial, previo a cualquier actividad.

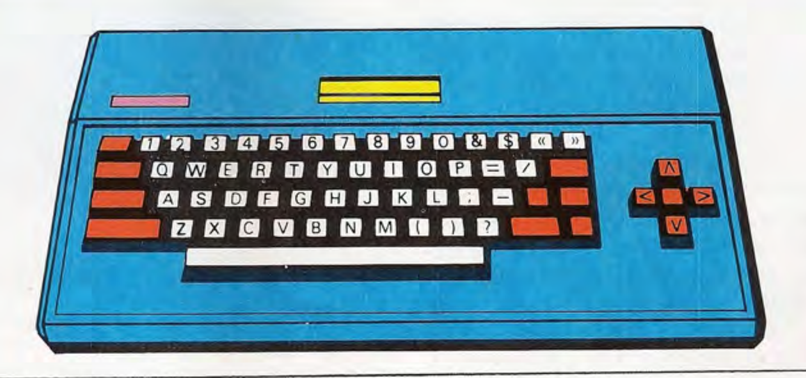
CLEAR (Limpiar pantalla): borra el contenido de la pantalla.

HOME: devuelve el cursor a la posición de origen.

SHIFT (Desplazamiento): selecciona el carácter superior de cada tecla; en las teclas alfabéticas, selecciona las mayúsculas correspondientes.

Desplazamiento del cursor: en el teclado existen cuatro teclas, cada una con la flecha en un sentido, que permiten mover el cursor en cualquier dirección.





bolo situado en el margen izquierdo de la pantalla (un cuadrado o un simple guión). Este es un elemento característico que señala la posición en la que se escribirá el próximo carácter que se lleve a la pantalla. Cualquier acción sobre una tecla tiene como consecuencia el desplazamiento del cursor en una posición a la derecha; su lugar lo irán ocupando las letras, cifras o símbolos que correspondan a las teclas pulsadas.

A pesar de que comparten una estructura semejante, el teclado de cada modelo de ordenador presenta sus particularidades. La forma de cada tecla y su distribución conjunta difiere de un ordenador a otro. Por lo demás, el número de teclas especiales, distintas de las teclas alfabéticas y numéricas de una máquina de escribir, también varía según el equipo.

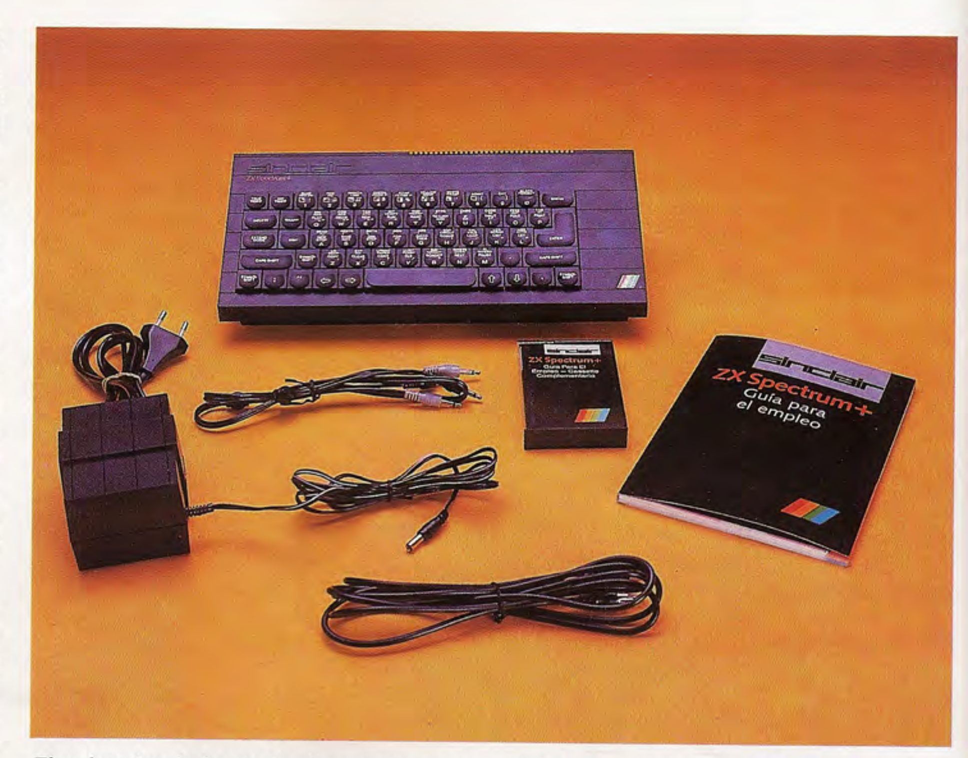
También difieren los caracteres y los símbolos que se introducen al accionar una determinada tecla simultáneamente con la de selección de mayúsculas («SHIFT»). Otro tanto ocurre con las órdenes y funciones seleccionadas al accionar una cierta tecla de caracteres junto con la tecla CONTROL («CTRL»). Estas diferencias no suponen un obstáculo que dificulte el diálogo con el ordenador. La única distorsión es que obligan al usuario a consultar el manual de su equipo, para saber de qué funciones dispone el teclado y con qué combinación de teclas se seleccionan.

Una vez que se conozca la forma de escribir caracteres en la pantalla, borrarlos, e insertar nuevos caracteres en cualquier posición, queda ya abierto el camino del diálogo. Este empezará con la escritura de las órdenes, instrucciones y mensajes que el usuario desee transmitir al ordenador; cada una de ellas, debe terminarla el usuario pulsando la tecla RETURN; o la tecla ENTER, su homóloga en determinados equipos. La acción sobre esta tecla indica al ordenador que debe «asimilar» el mensaje que la precede.

Es hora ya de estrenar el diálogo.

Una tras otra hemos pulsado las cuatro teclas y nuestro saludo aparece ya en la pantalla. Ahora hay que concluir la introducción accionando la tecla RETURN, para que la máquina asimile nuestro mensaje:

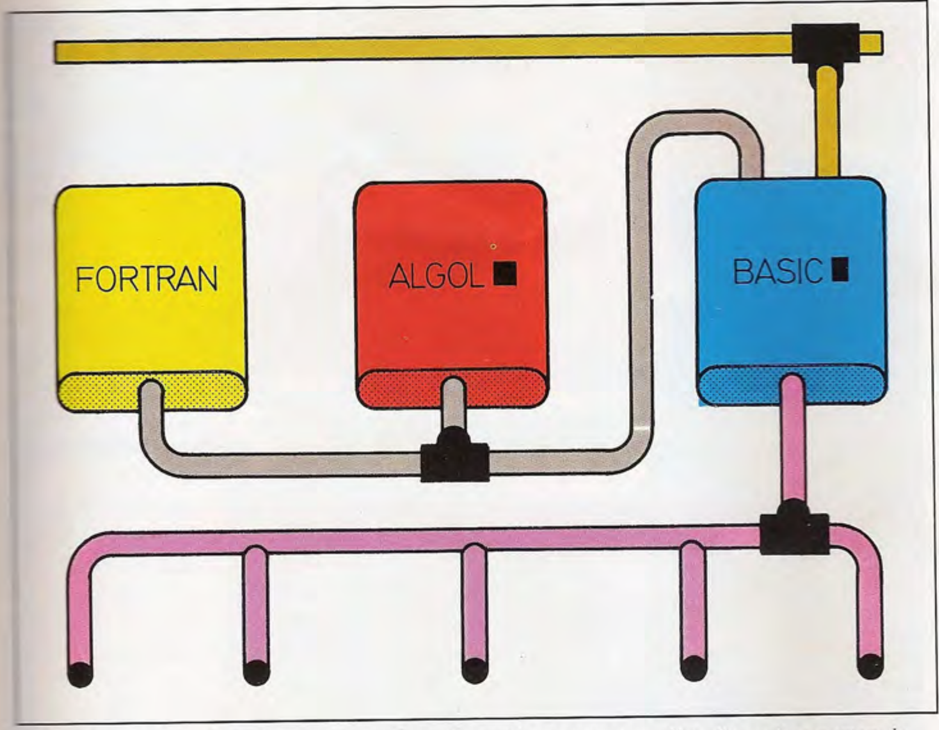
Aquí es donde se encuentra el verdadero obstáculo: la máquina no entiende el castellano. Hemos introducido el mensaje, seguido de una acción sobre la tecla RETURN... Sin errores. No obstante, la respuesta de la máquina es un mensaje de error: "SYNTAX ERROR" (o un mensaje parecido según el modelo de ordenador).



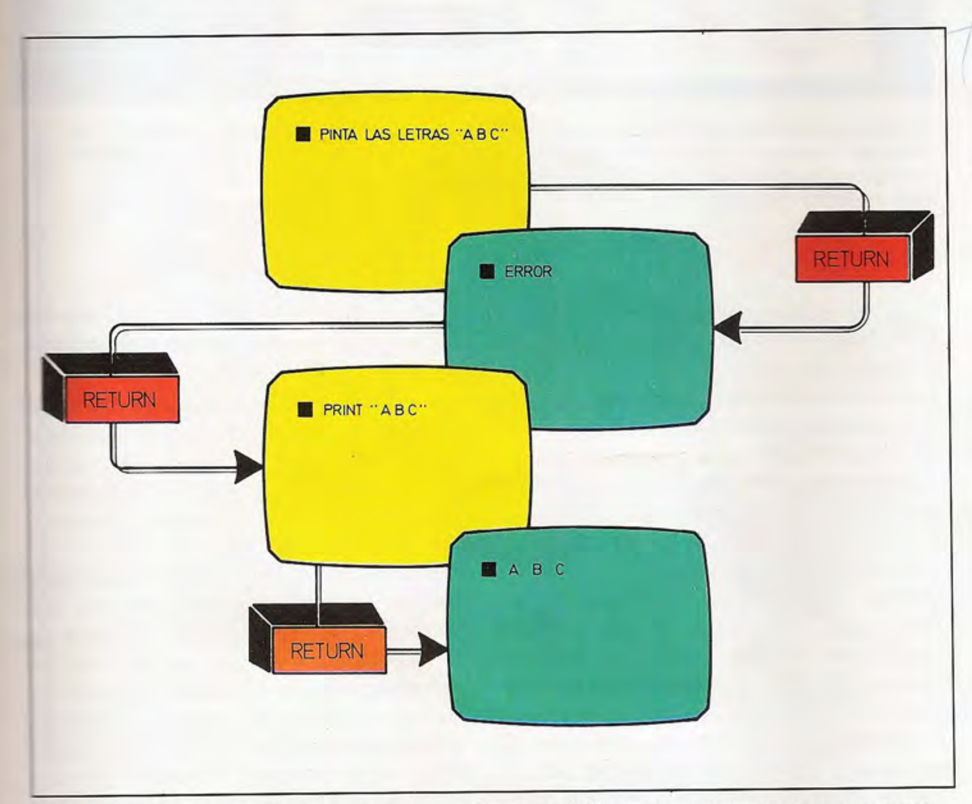
El ordenador, la fuente de alimentación, el manual de instrucciones y dos cables para unir la máquina con un magnetófono a casetes y con un receptor de TV. Todo lo necesario para tomar el primer contacto con la realidad del ordenador.



La principal vía a través de la que se establece el diálogo con el ordenador es el teclado. Este es semejante al de una máquina de escribir convencional, aunque con algunas teclas adicionales adecuadas para comunicar ciertas órdenes a la máquina.



El BASIC es el lenguaje de programación más popular en nuestros días. Sus raíces parten de otros dos lenguajes informáticos: el FORTRAN y el ALGOL.



Una vez conectado el ordenador, intentamos el primer diálogo. Con sorpresa observamos que no es capaz de entender nuestro idioma. En efecto, las máquinas programables tienen su propio lenguaje —el BASIC es el primordial—, que el usuario debe aprender y utilizar para hacer posible la comunicación con el ordenador.

El ordenador no entiende el mensaje "HOLA" y nos advierte que hemos cometido un error. En vista de los mensajes que muestra la pantalla, cabría suponer que su idioma es el inglés. Podemos intentarlo de nuevo introduciendo el mensaje traducido ("HELLO"). La respuesta sería idéntica: "SYNTAX ERROR".

En efecto, el ordenador no entiende ningún idioma «humano». Sólo es capaz de entender los mensajes escritos en determinados lenguajes especiales, los denominados *lenguajes de programación.* Además, su capacidad para dialogar en base a los lenguajes de programación no es general: sólo está capacitado para asimilar los mensajes formulados en un solo lenguaje de programación, aquel para el que se haya educado a la máquina.

Usted, por ejemplo, entiende perfectamente el castellano (su lengua materna), pero es capaz, bajo ciertas circunstancias (estudiando, viajando...), de entender inglés, francés... El ordenador personal tiene también su «lenguaje materno», que es el que se adjunta con el equipo; e incluso está capacitado para «aprender» otros lenguajes, en determinadas condiciones.

La conclusión es obvia: para establecer cualquier diálogo con el ordenador es necesario aprender su lenguaje de programación.

EL LENGUAJE BASIC

El lenguaje de programación más común en el campo de los ordenadores personales es el BASIC. Su nombre está construido a partir de las iniciales de *Beginner's All-Purpose Symbolic Instruction Code:* Código de instrucciones simbólicas de uso general para principiantes. Toda una definición del objetivo que motivó la creación de este popular lenguaje.

A la hora de desarrollar un lenguaje de programación polivalente y de fácil uso por los programadores noveles, los creadores del BASIC se inspiraron en dos lenguajes de alto nivel muy popularizados: el FORTRAN y el ALGOL.

Como tal lenguaje, el BASIC tiene su pro-

pio vocabulario y sus reglas sintácticas; y por supuesto, las particularidades inherentes a los lenguajes destinados al diálogo con los ordenadores.

Está concebido para ser una herramienta a la hora de utilizar el ordenador; en su mayor parte consta de órdenes que la máquina ha de ejecutar.

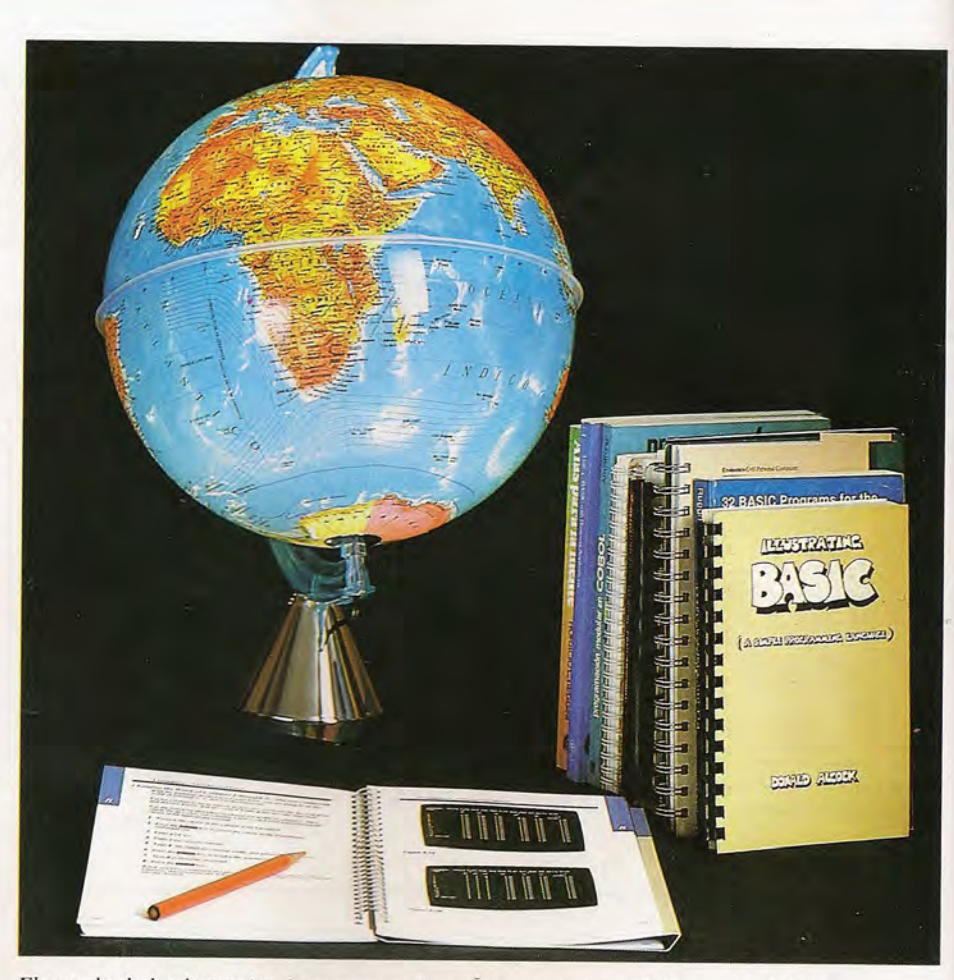
La pragmática de los lenguajes humanos se fundamenta en una unidad de comunicación: la frase. Combinando diversas fases es como se establece una conversación. Los diálogos en el lenguaje BASIC también se establecen a partir de frases, las *instrucciones,* que agrupadas ordenadamente dan cuerpo a un bloque de comunicación o *programa.*

A la hora de construir cada una de las instrucciones, el BASIC se acoge a una estructura semejante a la propia de las frases humanas. La combinación adecuada de sujeto, predicado y complementos, permitirá expresar una idea con detalle y concreción. Dentro del vocabulario del BASIC, caben verbos adecuados para definir las más diversas acciones que deban realizarse con los datos, los sujetos de las instrucciones. En síntesis, la tarea de programar al ordenador para que éste realice un trabajo, consistirá, sencillamente, en redactar una descripción detallada del conjunto de tareas a ejecutar. Esta descripción de tareas, o *programa,* constará de una secuencia ordenada de frases *(instrucciones),* cada una de ellas destinada a ordenar la ejecución de una de las tareas elementales. La rigidez del orden en el que deben ejecutarse las tareas que resuelven una actividad, se manifiesta en la numeración de las instrucciones dentro del programa.

Verbos como «imprimir», «asignar», «saltar», «leer», «ejecutar»... forman parte del vocabulario de este importante lenguaje de programación. Tan importante como el uso correcto del vocabulario, es el estricto cumplimiento de las reglas sintácticas. El ordenador no comprenderá una palabra en la que se haya cambiado una letra; igual que nosotros no entendemos el significado de «casa» si por un error ortográfico leemos «cosa». Al detectar un error sintáctico, el ordenador lo comunicará al usuario por medio del correspondiente mensaje.

La analogía con los lenguajes humanos llega hasta el punto de que también existen dialectos del lenguaje BASIC; dialectos condicionados por las particularidades del programa traductor específico que reside en cada ordenador.

Hay dialectos o versiones más potentes, con un vocabulario más completo, que permiten una mayor flexibilidad en la redacción de las instrucciones. También hay versiones más sencillas, con un vocabulario más reducido. A lo largo de la obra, se acometerá el estudio práctico del lenguaje BASIC más completo y generalizado para ordenadores personales; un estudio que se completará con una descripción de las variantes sintácticas y semánticas que caracterizan a los dialectos más importantes.



El mundo de los lenguajes de programación guarda un gran paralelismo con el de los lenguajes humanos. Existen idiomas de mayor relevancia, como el inglés, que constituyen verdaderos patrones de entendimiento universal. Esta característica se hace extensiva al ámbito de las máquinas, donde el BASIC se revela como el lenguaje más generalizado y universal.

Glosario

CONTROL DE PROCESOS: actividad en la que el ordenador se encarga de supervisar y controlar de forma automática determinados fenómenos o procesos externos.

SENSORES: dispositivos que pueden medir o detectar ciertos fenómenos físicos (temperatura, presión...)

UNIDAD DE DISCO FLEXIBLE: periférico que permite almacenar la información en un soporte físico externo denominado disco flexible. El soporte es un disco de plástico, sobre el que se ha depositado una capa de material magnetizable en el que se graba la información.

UNIDADES DE ALMACENAMIENTO: dispositivos que permiten almacenar la información manipulada por el ordenador, para su posterior recuperación y tratamiento.

PROGRAMA: conjunto ordenado de instrucciones codificadas en un lenguaje de programación que comunican al ordenador una tarea a realizar.

Lenguajes informáticos

Del lenguaje máquina a los lenguajes de alto nivel

Ocurre en el universo de los seres humanos y, por supuesto, también debía suceder algo semejante en el mundo de los ordenadores. Las máquinas programables creadas por el hombre, ocupan una torre de Babel en la que la disparidad de los lenguajes es tan acusada como pueda serlo en la sociedad humana. Al igual que en el terreno de los lenguajes humanos hay algunos idiomas con mayor relevancia y difusión, utilizados como patrón de entendimiento (el inglés, el castellano o el francés), en la Babel de los ordenadores también existen algunos lenguajes con acusado protagonismo. Sin lugar a dudas, el más popular de ellos es el BASIC

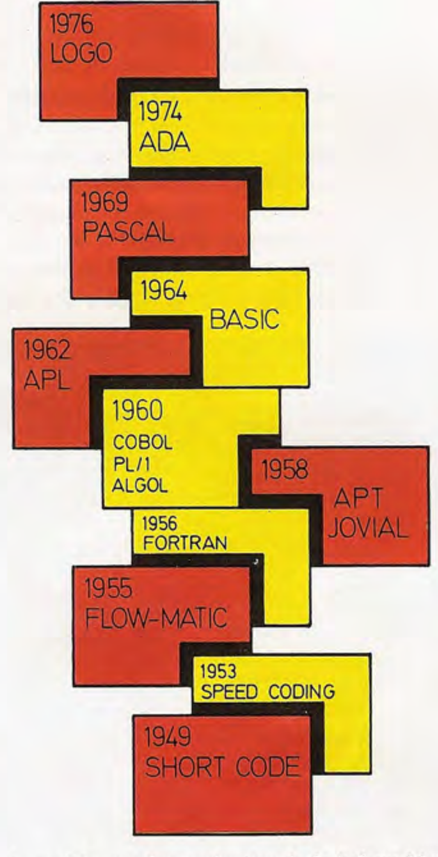
No terminan aquí las analogías. En la sociedad humana, muchos idiomas presentan dialectos o variantes, más o menos distantes de la raíz original. Una realidad trasladable a los lenguajes informáticos, donde los dialectos y versiones del BASIC son casi tan numerosos como modelos de ordenadores.

Cualquier exposición relativa a los lenguajes informáticos, debe partir del personaje central: el ordenador. Sabemos que es una máquina cuya propiedad diferenciadora reside en que admite una programación que le permitirá realizar una ingente variedad de tareas: tantas como el usuario sea capaz de programar.

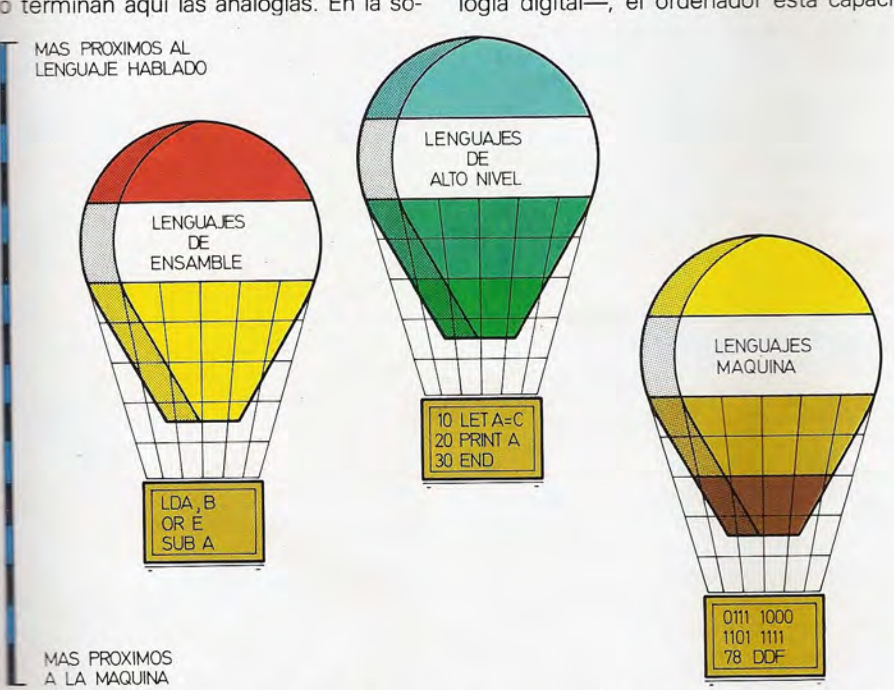
Debido a su constitución íntima —un conjunto de circuitos electrónicos, de tecnología digital—, el ordenador está capacitado para entender un lenguaje construido a partir de dos elementos mínimos de información: 0 (ausencia de señal eléctrica) y 1 (presencia de señal eléctrica).

A partir de este alfabeto mínimo (0 y 1), pueden construirse palabras (01101110) y, por supuesto, frases (00110011 11100011...) con las que establecer una comunicación repleta de contenido.

Algo es ya evidente: por su naturaleza, el ordenador está capacitado para dialogar



A medida que ha evolucionado la informática han ido creándose nuevos lenguajes de programación, «de alto nivel», con una estructura semejante a la de los lenguajes humanos.



El ordenador es una máquina capaz de recibir una programación y comunicarse con el mundo exterior. Ello es posible debido a la existencia de lenguajes que permiten establecer una comunicación organizada; éstos son los denominados lenguajes de programación, clasificables en tres categorías según sean más o menos próximos al lenguaje íntimo de los circuitos electrónicos de la máquina.

en un lenguaje íntimo que denominaremos *lenguaje máquina,* edificado a partir de dos elementos mínimos de información: 0 y 1. Estos elementos, los *BITs* (contracción de su denominación inglesa Binary DigiT: dígito binario), coinciden con los propios del sistema de numeración binario.

NIVELES DE LOS LENGUAJES INFORMATICOS

La capacidad del ordenador para intercambiar información con el usuario es un hecho ya precisado. Por el momento, hemos visto que es posible establecer una comunicación con sus circuitos electrónicos utilizando su lenguaje íntimo: el lenguaje máquina. Una simple reflexión acerca de esta vía de diálogo, nos lleva a algunas conclusiones no excesivamente favorables para el lenguaje máquina. En principio, no cabe duda que se trata de un lenguaje difícil de aprender para el usuario; no sólo por la complejidad inherente a la estructura del propio lenguaje, sino también por el hecho de que su íntima relación con la máquina obliga a conocer muy a fondo sus entresijos. Además, hay que recordar que el ordenador es una máquina ignorante, a la que hay que instruir con toda suerte de detalles mínimos; una tarea ardua y difícil de llevar a buen término a base de combinar ceros y unos.
Al fin y al cabo, el ordenador no es más que una herramienta creada por el hombre para facilitar su trabajo... ¿Por qué hay que acondicionarse al lenguaje íntimo de la máquina, si cabe la posibilidad de comunicarse con ella utilizando un lenguaje próximo al humano? Todo el problema se reduciría a crear los adecuados traductores, que convirtiesen las descripciones formuladas en lenguaje evolucionado en sus homólogas en lenguaje máquina.



La disparidad de lenguajes y la necesidad de medios de traducción que permitan el entendimiento, son realidades compartidas por los lenguajes humanos convencionales y los lenguajes de programación.

Los principales lenguajes de alto nivel

ADA
(En honor de Lady Augusta ADA Byron)

La publicación de unas notas sobre la máquina analítica de Charles Babbage, le sirvió a la condesa de Lovelace para pasar a la posteridad, cediendo su nombre al lenguaje que debía nacer del proyecto GREEN. Fue en 1975 cuando se consumaron los trabajos del equipo dirigido por J. M. Ischbia de la firma CII-Honeywell Bull, con el patrocinio del Departamento de Defensa de los Estados Unidos.
El ADA es un lenguaje inspirado en el PASCAL que se desarrolló con el objetivo de conseguir un lenguaje con posibilidades de convertirse en estándar universal y que facilitara el mantenimiento de los programas. Actualmente, aún no está muy difundido, aunque muchos expertos lo consideran uno de los lenguajes con mayor futuro.

ALGOL
(ALGOrithmic Language. *Lenguaje algorítmico)*

A raíz de un proyecto de Peter Naur en 1958, un consorcio internacional promovió el desarrollo de un lenguaje de alto nivel, inicialmente para aplicaciones científicas, que algún tiempo más tarde se plasmaría en el ALGOL. A pesar de sus cualidades para cálculos numéricos, tratamiento de entradas/salidas y procesos recursivos, el ALGOL es uno de los lenguajes que no ha viajado con la revolución microinformática.

BASIC
(Beginners All-purpose Symbolic Instruction Code. *Código de instrucciones simbólicas de uso general para principiantes)*

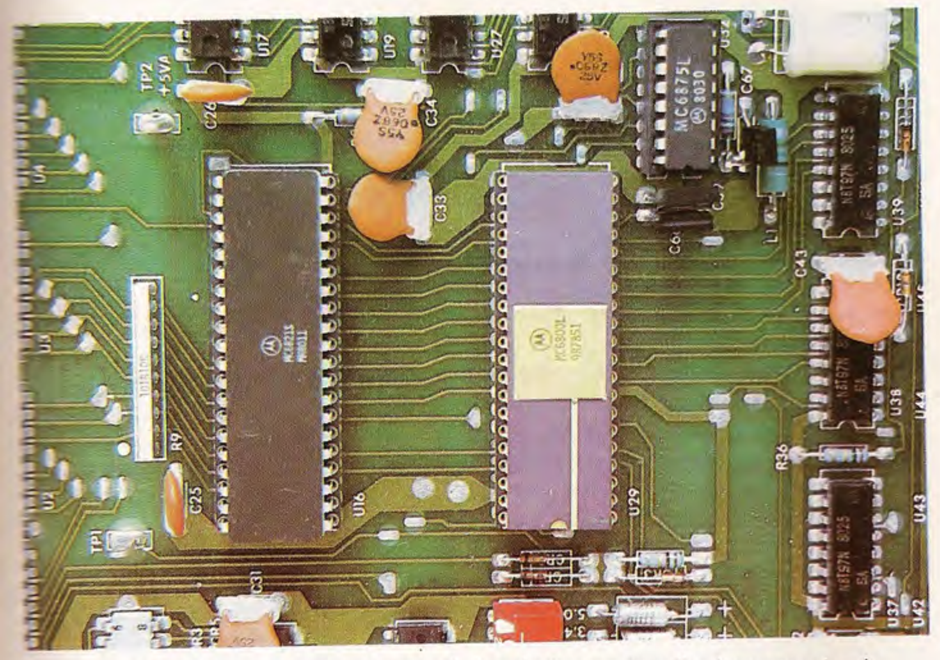
El más popular de los lenguajes actuales, sin lugar a dudas, y a considerable distancia de los restantes lenguajes informáticos. Nació entre 1964 y 1965 en el Dartmouth College como una herramienta para la enseñanza. Con el tiempo, han ido proliferando los dialectos y versiones, hasta el punto de que raro es el fabricante que no desarrolla un dialecto propio para sus equipos.
Es muy difícil encontrar un ordenador personal que en su versión básica no incorpore un intérprete de lenguaje BASIC. Desde hace algunos años, la firma americana Microsoft lidera el desarrollo de dialectos BASIC, disponiendo de algunas versiones (MBASIC, MSX-BASIC...) que se están convirtiendo en núcleos de estandarización.

«C»

Uno de los más recientes lenguajes de programación de alto nivel. Dentro de este marco, el «C» es uno de los lenguajes más polivalentes y próximos a la realidad de la máquina. La Bell Laboratories lo desarrolló en su origen para trabajar con el sistema operativo UNIX. La popularidad del «C» crece día a día; ello permite catalogarlo como uno de los lenguajes del futuro. Su estructura sintáctica y semántica está edificada sobre conceptos tales como estructuración, jerarquización de bloques y control del flujo de datos.

COBOL
(COmmon Business Oriented Language. *Lenguaje orientado a la gestión)*

Sin lugar a dudas, se trata del lenguaje especializado en tareas de gestión que ha alcanzado una mayor resonancia. El Departamento de Defensa de los Estados Unidos, promovió su desarrollo en 1960. A



En su intimidad, el lenguaje de los circuitos de la máquina son complicadas secuencias de niveles de tensión: estados de conexión y desconexión que representamos por medio de combinaciones de ceros y unos.

Un razonamiento que derivó en la creación de nuevos lenguajes informáticos más próximos al lenguaje convencional.
El desarrollo de los lenguajes fue paralelo a la evolución de los ordenadores. Poco a poco, fueron naciendo lenguajes algo más alejados de la intimidad de la máquina y, más tarde, llegaron los denominados *lenguajes de alto nivel,* cuya sintaxis, semántica y pragmática eran ya semejantes a los del lenguaje humano.
En nuestros días, la pirámide de los lenguajes informáticos consta de tres niveles:

- *Lenguajes máquina.*

Ocupan el estrato inferior, menos evolucionado, de los lenguajes informáticos. Dada su total consonancia con la naturaleza íntima de la máquina, es obvio que el lenguaje será distinto según se trate de una u otra máquina.
Cabe recordar que el cerebro o unidad central del proceso es el microprocesador; en consecuencia, el lenguaje má-

pesar de las críticas formuladas por algunos teóricos, expertos en lenguajes informáticos, su presencia es aún frecuente en miniordenadores y grandes equipos. No ocurre lo mismo en el terreno de los microordenadores; su representación en este campo se reduce a algunas versiones compiladas compatibles con el sistema operativo CP/M.

FORTH

Un lenguaje difícil de catalogar, dada su distancia respecto a los restantes lenguajes de alto nivel. A pesar de acogerse al concepto de estructuración, el FORTH mantiene una acusada proximidad respecto al lenguaje máquina; con las contrapartidas que ello supone, en cuanto a velocidad de ejecución y reducida ocupación de memoria. Otra característica reseñable es su evolutividad; el FORTH permite al usuario crear sus propios comandos. Día a día crece su proyección en el ámbito de los microordenadores. En la actualidad existen compiladores o semi-compiladores FORTH para un gran número de ordenadores personales.

FORTRAN
(FORmula TRANslation. *Conversión de fórmulas)*

Su nombre evidencia la orientación matemática de uno de los más antiguos de los lenguajes que aún predominan en nuestros días. J. Backus lo desarrolló en 1956 sobre un ordenador IBM 704. A pesar de su orientación primaria, el FORTRAN se ha revelado como un lenguaje adecuado para aplicaciones de gestión.
Aunque ha perdido terreno frente a lenguajes más modernos, persiste su empleo de la mano de compiladores con versiones compatibles con sistemas operativos tan populares como el CP/M.

LISP
(LISt Processing. *Procesado de listas)*

El Massachusetts Institute of Technology creó en 1959 este lenguaje de alto nivel orientado a aplicaciones de inteligencia artificial. La programación de procesos recurrentes (edificados sobre datos sintetizados en los pasos anteriores) es uno de los puntos fuertes del LISP.
Dentro de su especialidad, es un lenguaje que sigue en plena vigencia, y del que existen compiladores para microordenadores y ordenadores personales.

LOGO

Seymour Papert, del Massachusetts Institute of Technology, creó en 1976 la primera versión del popular LOGO, inspirada en su anterior desarrollo: el lenguaje LISP.
El LOGO es un lenguaje especialmente adecuado para la enseñanza asistida por ordenador. Su celebridad se debe en gran parte a la simpática «tortuga»: el símbolo con cuyo desplazamiento se generan los dibujos y presentaciones gráficas. A pesar de su popularidad, es un lenguaje encasillado en el campo educativo. Permanece ignorado por los profesionales de la programación, aunque no es desdeñable su utilidad como herramienta para la simulación de fenómenos de inteligencia artificial.

PASCAL
(En honor del célebre matemático francés Blaise PASCAL)

Este es el lenguaje estructurado por excelencia, con una presencia más que importante en el mundo de los microordenadores. N. Wirth lo desarrolló en 1969, en la escuela politécnica de Zurich, partiendo de los fundamentos del ALGOL. El PASCAL es un lenguaje muy adecuado para generar programas comprensibles y claros; ello se debe a su característica de lenguaje estructurado que obliga a la definición previa de todos los parámetros en juego.
La Universidad Californiana de San Diego, desarrolló la versión de PASCAL más popular en el campo de los microordenadores y ordenadores personales: el PASCAL UCSD.

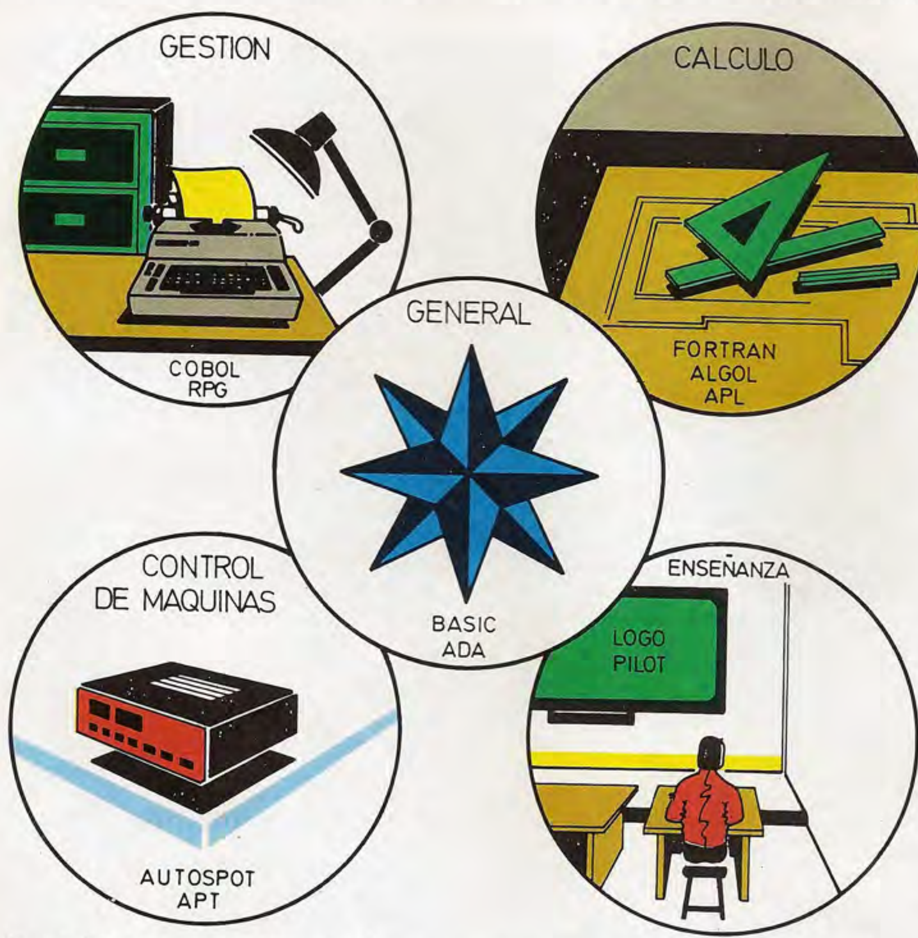
quina apropiado para cada ordenador dependerá del tipo de microprocesador que resida en su núcleo. En efecto, cada microprocesador (6800, 6502, Z80...) tiene su propio lenguaje máquina.

- *Lenguajes de ensamble o ensambladores.*

El estrato intermedio está ocupado por los lenguajes de ensamble. El repertorio de elementos que intervienen en la confección de los programas coincide, en este caso, con conjuntos de símbolos o nemónicos, que ofrecen una mayor comodidad que las asociaciones de ceros y unos.
Su relación con el lenguaje máquina es muy próxima, hasta el punto de que cada familia de microprocesadores posee un lenguaje ensamblador propio, en directa correspondencia con su lenguaje máquina.
La tarea de confección y corrección de los programas resulta ahora más fácil, dada la comodidad que supone el emplear grupos de letras en lugar de ceros y unos para definir las operaciones. No cabe duda que para incrementar un número en una unidad, es más grato escribir INC A que 00111100; y, desde luego, la posibilidad de cometer un error es bastante más reducida.

- *Lenguajes de alto nivel.*

Estos son ya lenguajes evolucionados que mantienen un gran paralelismo con los lenguajes hablados convencionales. En este tercer nivel, la disparidad de los lenguajes no es achacable al microprocesador. Los lenguajes de alto nivel más difundidos (BASIC, PASCAL, FORTRAN, COBOL, LOGO...) disponen de traductores para su conversión al lenguaje máquina de casi cualquier microprocesador.
Las ventajas son evidentes: la redacción del programa resulta comprensible para el usuario y, por lo tanto, es más cómoda su redacción y la detección de posibles errores sintácticos. Por lo demás se reduce el tiempo de programación y, lo que es más importante, cabe ya pensar en que un mismo programa pueda ser ejecutado por distintos ordenadores.



Dentro del universo de los lenguajes de programación, caben desde lenguajes orientados al diálogo especializado (gestión, cálculo, enseñanza...), hasta lenguajes polivalentes útiles para programar cualquier tipo de aplicación.

LENGUAJES DE ALTO NIVEL

El BASIC es, sin lugar a dudas, el lenguaje de alto nivel más popular en nuestros días. Un lenguaje de programación polivalente, que ha derivado en múltiples dialectos: desde versiones resumidas para el aprendizaje, hasta potentes y evolucionadas versiones orientadas a la programación profesional. Pero no arranca de ahí la historia de los lenguajes informáticos. Antes del BASIC y después del BASIC son muchos los lenguajes que han visto la luz: desde el SHORT CODE que en 1949 desarrolló el Doctor Mendy para la firma UNIVAC, o el SPEED CODING nacido en 1953 con destino a IBM, hasta los más recientes como el LOGO (1976), un lenguaje especialmente diseñado para la enseñanza asistida por ordenador.
Los primeros lenguajes se desarrollaron pensando en aplicaciones matemáticas y científicas: tal es el caso del MATHMATIC, el GAL o el tradicional FORTRAN. Más tarde llegaron los lenguajes orientados a la programación de aplicaciones de gestión, cuyo protagonismo corresponde al COBOL.
Los lenguajes polivalentes o universales llegaron tras la estela del JOVIAL, el primer lenguaje no especializado que vio la luz en 1959. A éste le siguieron el ALGOL, PL/1, APL, BASIC, PASCAL, FORTH, ADA, LOGO y otros muchos más.
La popularidad de los ordenadores personales ha arrastrado a algunos lenguajes de alto nivel; si bien, a su vez, ha relegado a otros lenguajes clásicos al ámbito de los miniordenadores y ordenadores gigantes.
Actualmente, tras el liderazgo del BASIC con sus múltiples dialectos, se alinea una corte de alternativas cuyos representantes más significados son: el PASCAL, LOGO, ADA, «C», FORTH y PILOT, además de ciertas versiones actualizadas de los tradicionales COBOL (CIS COBOL y COBOL-80), FORTRAN (FORTRAN 77, FORTRAN 80, FORTRAN SSS), LISP (MULISP) y PL/1 (con sus versiones PL/M, PL/W, PL/Z...).

El sistema operativo

La inteligencia elemental del ordenador

El símbolo más representativo de la época en que vivimos es una máquina llamada ordenador. Su aspecto externo, e incluso su interior —una amalgama de circuitos electrónicos—, es semejante al de muchos otros aspectos especializados en una determinada tarea. Equipos de alta fidelidad, receptores de radio y de TV, y otros sofisticados instrumentos electrónicos, están construidos a base de componentes y dispositivos que también se encuentran en la intimidad del ordenador. ¿Cuál es, entonces, la línea divisoria entre el mundo de las máquinas especializadas y del ordenador?

La distinción esencial entre el ordenador y cualquier otra máquina reside en que el primero es una máquina programable. A diferencia con cualquier otro sofisticado aparato de los que inundan los hogares, oficinas, colegios, industrias..., el ordenador es una máquina que el usuario puede instruir para que realice una tarea específica y distinta en cada ocasión.



La línea divisoria entre el ordenador y cualquier otra sofisticada máquina electrónica, la establece la posibilidad de recibir una programación. El ordenador es una máquina programable, con la que puede establecerse una comunicación y a la que es posible instruir para que realice la actividad que desee el usuario.

LA INTELIGENCIA ELEMENTAL DEL ORDENADOR

La tarea de «educación» de la máquina, se verá facilitada en gran medida si ésta cuenta con una inteligencia básica que agilice la comunicación entre el ordenador y el usuario. Semejante capacidad elemental debe brindar al usuario los medios adecuados para que éste controle y explote las posibilidades del ordenador.

La inteligencia elemental con la que hay que dotar a la máquina debe sintetizar tres grupos de funciones o capacidades básicas:

- Crear el entorno adecuado para el diálogo hombre/máquina. Tarea que supone el control de los dispositivos periféricos a través de los que se establece la comunicación entre el usuario y el ordenador: teclado, pantalla de visualización, impresora...

- Gestionar de forma automática la lectura y el almacenamiento de información (programas y datos) en las unidades de memoria que forman parte del sistema ordenador: unidad de cinta, de disco...

- Ofrecer al usuario los medios adecuados para el tratamiento de los archivos e información y para el conocimiento de su estado y situación en cualquier instante.

El cometido de la *inteligencia básica* del ordenador es evitar, en definitiva, la com-

pleta programación del hardware de la máquina cada vez que el usuario se decida a utilizarla.

EL SOFTWARE DEL SISTEMA

Dada la naturaleza del ordenador, una dualidad circuitería física/programación (hardware/software), resulta evidente que al hablar de dotarlo de una inteligencia básica, estamos apuntando a un elemento software que, de modo permanente, instruya a la máquina y la ponga en situación de entablar un diálogo con el exterior.

La «educación» de la máquina para realizar cada una de las funciones necesarias, correrá a cargo de un determinado número de programas. En su conjunto estos programas, que denominaremos *software del sistema,* constituyen la inteligencia básica del ordenador.

No está de más recordar que el hardware debe ser permanentemente instruido, hasta el más mínimo detalle, para que pueda manifestar su capacidad en el tratamiento de información (objetivo de cualquier ordenador).

En los ordenadores más pequeños, son escasas las funciones del software del sistema. Su actuación se manifiesta, únicamente, en funciones tales como instruir al ordenador para que detecte una acción sobre el teclado, identifique cuál ha sido la tecla pulsada y lleve su valor a la pantalla al tiempo que lo almacena en la memoria.

También suelen ofrecer al usuario la posibilidad de examinar la información almacenada en determinadas zonas de la memoria, e incluso, de observar cuál es el estado de algunos registros internos.

A medida que el ordenador es más potente y evolucionado, crecen las posibilidades de su inteligencia elemental. En la actualidad, ordenadores personales suelen poseer un software del sistema que ofrece al usuario posibilidades más que notables. Permiten a éste encomendar a la máquina muy diversos tipos de acciones, sin más que ordenarlo introduciendo el comando al efecto.

En los equipos más pequeños, el software del sistema adopta la forma de un programa, grabado permanentemente en su memoria. Este recibe un nombre tan ilustrativo como es el de *programa monitor.* Cuando el ordenador es ya una máquina evolucionada, como es el caso de un ordenador personal para aplicaciones profesionales o de gestión, el software del sistema consta de todo un «paquete» de programas que obedece a la denominación genérica de *Sistema Operativo* (S.O. al reducirlo a las iniciales).



Toda la eficacia práctica del ordenador depende de la perfecta coexistencia de dos elementos indisolubles: el «hardware» o circuitería física y el «software» o componentes de la programación.



Los modernos ordenadores poseen una inteligencia elemental cada vez más evolucionada. Incorporan potentes sistemas operativos que brindan al usuario toda la eficacia práctica del hardware de la máquina.

¿QUE ES UN SISTEMA OPERATIVO?

Al extraer las conclusiones de los apartados precedentes, surge la definición de sistema operativo:

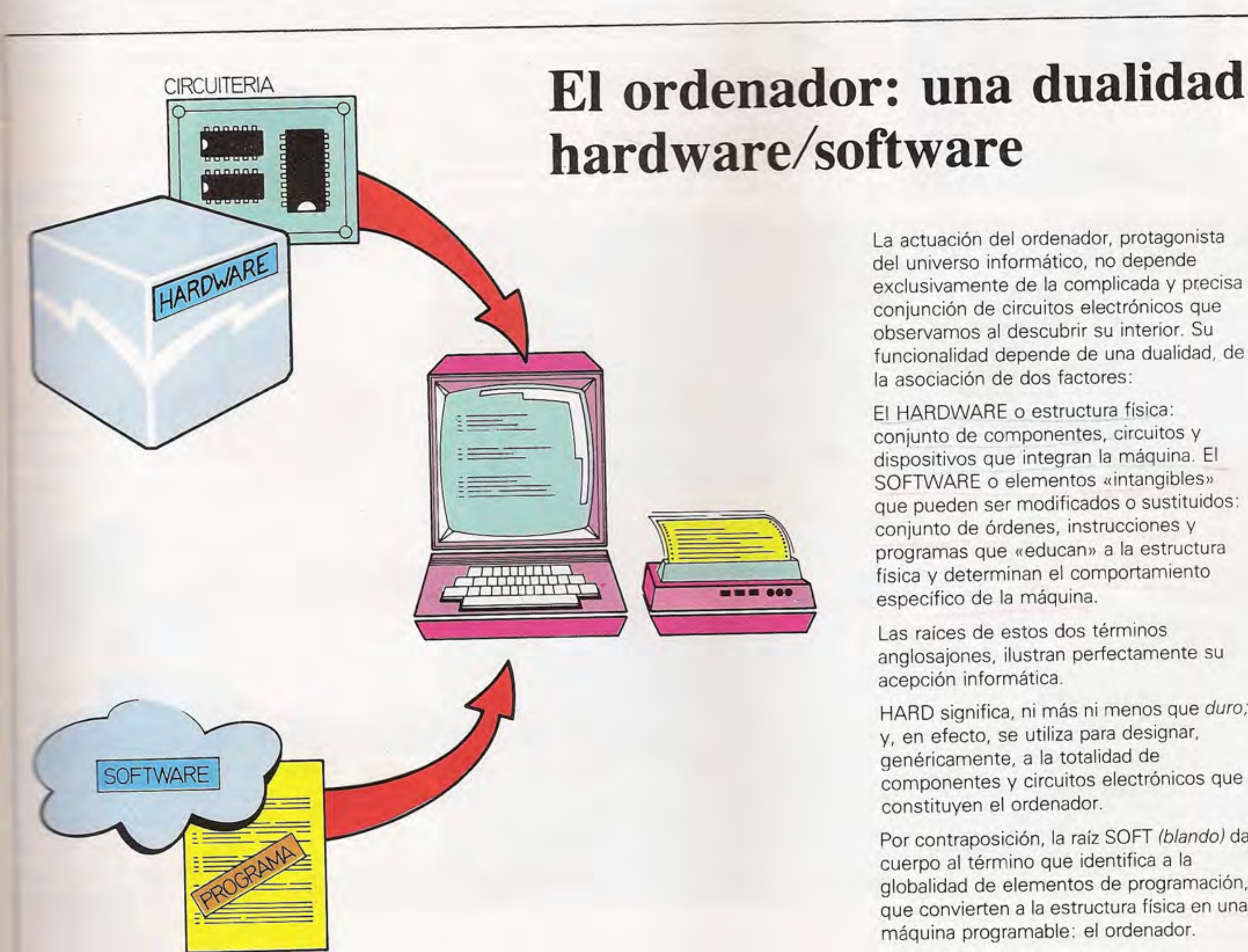
Conjunto de programas que constituyen

la inteligencia básica del ordenador y cuya misión es crear el marco adecuado para una eficaz comunicación entre el ordenador y el usuario o mundo exterior.
Con la llegada del microprocesador, los ordenadores han reducido su tamaño hasta el punto de que cualquier moderno microordenador u ordenador personal puede competir, e incluso superar, a sus antepasados, docenas de veces más voluminosos y lentos. La revolución no sólo se manifiesta en la vertiente hardware de los equipos, sino también en el software, y no iba a ser menos en el terreno de los sistemas operativos.
Dos son las consecuencias de la evolución de los sistemas operativos:

- *El constante avance en la potencia y capacidad de tratamiento.* Cada vez son mayores las posibilidades de los sistemas operativos: pueden controlar a un mayor número de periféricos asociados al ordenador, ponen a disposición del usuario un repertorio de comandos más amplio y potente y, un factor de gran importancia, ponen en práctica nuevos métodos, más eficaces y rápidos, para explotar las posibilidades del hardware de la máquina.

- *Una acusada tendencia hacia la estandarización.* Hasta hace algunos años, la disparidad de los sistemas operativos era casi absoluta. A medida que creció el número de ordenadores, se avanzó en la idea de que compartieran un mismo cerebro circuital (el procesador o CPU). La llegada del microprocesador, y su implantación como cerebro de los microordenadores, supuso un definitivo avance en el camino hacia la estandarización de los sistemas operativos.

En la actualidad son muy pocos los tipos de microprocesadores que ostentan el liderazgo del «mercado de cerebros» para microordenadores. Z-80, 6502, 6809, 8086, 8088 y 68000, son los nombres de estos microcerebros que, alojados en una superficie no mayor que un sello postal y recubiertos por una simple cápsula de plástico o de cerámica, están dispuestos a ejecutar con premura y eficacia las órdenes que reciban del mundo exterior.
Semejante uniformidad ha hecho posible que un mismo sistema operativo pueda ser compartido por una amplia variedad de ordenadores, cuya CPU está basada en un mismo tipo de microprocesador.
Desde luego, el camino hacia la estandarización es aún incipiente. Hoy siguen siendo múltiples los sistemas operativos que coexisten en el mercado, con una mayor o menor implantación. Si bien, un



El ordenador: una dualidad hardware/software

La actuación del ordenador, protagonista del universo informático, no depende exclusivamente de la complicada y precisa conjunción de circuitos electrónicos que observamos al descubrir su interior. Su funcionalidad depende de una dualidad, de la asociación de dos factores:

El HARDWARE o estructura física: conjunto de componentes, circuitos y dispositivos que integran la máquina. El SOFTWARE o elementos «intangibles» que pueden ser modificados o sustituidos: conjunto de órdenes, instrucciones y programas que «educan» a la estructura física y determinan el comportamiento específico de la máquina.

Las raíces de estos dos términos anglosajones, ilustran perfectamente su acepción informática.

HARD significa, ni más ni menos que *duro;* y, en efecto, se utiliza para designar, genéricamente, a la totalidad de componentes y circuitos electrónicos que constituyen el ordenador.

Por contraposición, la raíz SOFT *(blando)* da cuerpo al término que identifica a la globalidad de elementos de programación, que convierten a la estructura física en una máquina programable: el ordenador.

reducido grupo de ellos —los sistemas operativos CP/M y MS/DOS, básicamente— ocupan un liderazgo destacado, revelándose como protagonistas del avance hacia la estandarización.

FUNCIONES DE UN SISTEMA OPERATIVO

La presencia del sistema operativo en los ordenadores responde a dos objetivos básicos. El primero no es otro que convertir a la máquina en un ordenador, practicable y eficaz, con capacidad para iniciar un diálogo con el mundo exterior. De esta situación parte el segundo de los objetivos del S.O.: explotar al máximo los recursos y posibilidades del *hardware* del ordenador para que su uso sea el óptimo.
La puesta en práctica de ambos objetivos básicos, exige al sistema operativo una notable capacidad de gestión y proceso. Capacidad que se resume en tres niveles funcionales compartidos por cualquier sistema operativo evolucionado.

- *Gestión del propio sistema ordenador,* lo que equivale a supervisar y controlar tanto el funcionamiento de la unidad central, como el de las unidades periféricas asociadas (pantalla, teclado, impresora, unidades de almacenamiento...).

- *Gestión de los trabajos encomendados a la máquina.* El control y tratamiento de las tareas que se le han encomendado, exige al sistema operativo capacidad para:
— Planificar los trabajos, respetando las prioridades que pudieran haberse otorgado a cada uno de ellos.
— Asignar los recursos de la máquina para la eficaz resolución de las tareas a procesar. Ello se traduce en la asignación y reserva de zonas de memoria, dedicación de periféricos adecuados para cada actividad y control de los mismos...
— Supervisar y establecer las comunicaciones oportunas con el entorno, tanto para la carga de programas y datos, como para entregar los resultados al exterior.

- *Gestión de datos.* Con toda la actividad que conlleva la estructuración de archivos, el acceso a los mismos, el control de los soportes de memoria externa y la propia verificación y manipulación de los datos.
Un repaso a las funciones que incorpora el sistema operativo, revela su perfecta adecuación a las exigencias que se imponen a la «inteligencia elemental» de la máquina.

En efecto, las tres funciones básicas d[e] un S.O. contribuyen a crear el entorn[o] adecuado para la comunicación hombr[e] máquina, gestionan la lectura y el almac[e]namiento automático de la informació[n] (programas y datos) y, por último, ofrece al usuario los medios adecuados para [el] tratamiento de los archivos e informan d[el] estado del sistema en cualquier instante.

El software del sistema

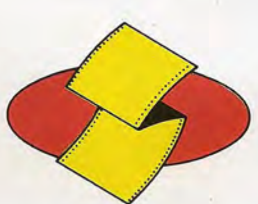
Por su constitución física, hemos visto que el ordenador es una simple máquina; no más compleja que cualquiera de los sofisticados instrumentos electrónicos y equipos domésticos que han asaltado los hogares. Un entramado de circuitos electrónicos y dispositivos mecánicos de precisión que puede convertirse en un ordenador, capaz de recibir una programación y realizar innumerables funciones.

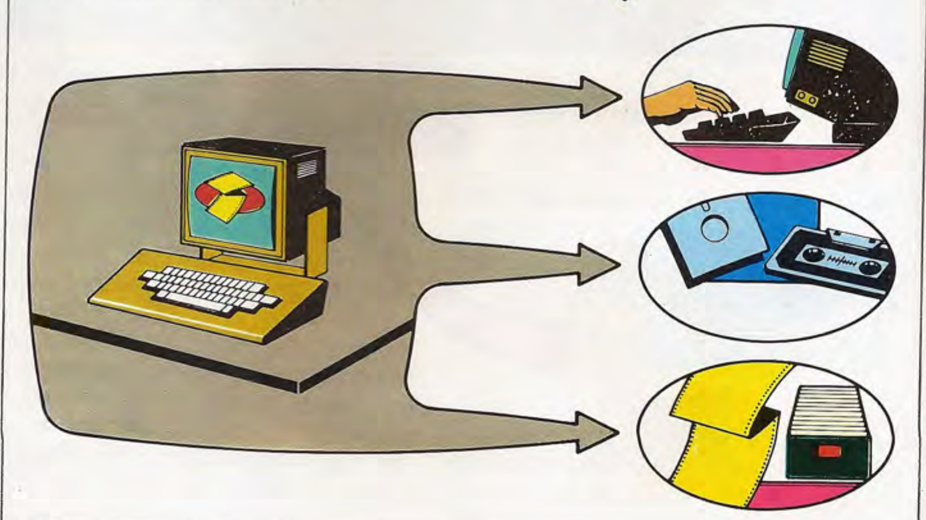
El salto de máquina a ordenador es posible desde el trampolín que aporta a la máquina el software del sistema. Esta inteligencia elemental instruye a la amalgama de circuitos electrónicos para que puedan llegar a establecer una comunicación con el mundo exterior. De esta forma, la máquina puede recibir una programación o *educación* para realizar las tareas que el usuario desee.

Tres son las misiones básicas encomendadas al software del sistema:

- Crear el entorno adecuado para la comunicación hombre/máquina.
- Gestionar la lectura y el almacenamiento de información en las unidades de memoria.
- Ofrecer al usuario los medios adecuados para el tratamiento de los archivos de información y para el conocimiento del estado del ordenador en cualquier instante.







Software de aplicación

El último nivel del edificio informático

Según los expertos, más del cincuenta por ciento del mercado informático está ocupado por el *software*. Un concepto que engloba a la totalidad de elementos que intervienen en la educación del *hardware* o circuitería física de la máquina, y que convierten a ésta en un verdadero ordenador.
Ambos aspectos, hardware y software, son indisolubles. Su complementariedad es la que determina la existencia de la informática.

¿BASTA CON UN S.O. Y UN TRADUCTOR DE LENGUAJE?

Partiendo de la máquina desnuda, del hardware, hay que llegar a construir un ordenador con toda su capacidad y posibilidades prácticas. Semejante edificio, consta de varios niveles. En una primera etapa, hay que revestir al hardware de lo que denominaremos la «inteligencia básica»: el sistema operativo. Sobre este primer nivel, es posible ya implantar el traductor de lenguaje que permitirá establecer un diálogo organizado con la máquina.
La transformación de la máquina en la útil herramienta práctica que conocemos, se completa al ocupar el tercer nivel del edificio: al dotar al ordenador del *software de aplicación*. Este puede estar constituido, sencillamente, por programas confeccionados por el usuario en el lenguaje que es capaz de reconocer el traductor. Programas de juego, programas adecuados para realizar cálculos matemáticos o para gestionar la agenda telefónica personal... O incluso programas más desarrollados, destinados a tareas bastante más complejas: contabilidad doméstica, archivo bibliográfico personal, control de movimientos de las cuentas bancarias...
La gestión de aplicaciones complejas exige la presencia en el ordenador de un software de aplicación especializado, complejo y optimizado.



De máquina a ordenador. Una estructura a edificar sobre los cimientos del hardware y que, al completar los sucesivos niveles, dará lugar al nacimiento del ordenador: un disciplinado y eficaz colaborador en cualquier actividad.



Para dar a la máquina toda su eficacia práctica, no basta con dotarla de un sistema operativo y un traductor de lenguaje. La «instrucción» para realizar cualquier actividad la recibe de la mano de los programas de aplicación.

Una actividad tan habitual como puede ser la gestión de un archivo de clientes, se ve facilitada y potenciada al sustituir los métodos tradicionales de archivo por un ordenador, dotado de un software de aplicación al efecto. Este consistirá en un conjunto o «paquete» de programas (de ahí la denominación de «paquetes de aplicación»). Cada uno de los programas, destinados a resolver una tarea específica (apertura de ficha a nuevos clientes, relación de clientes con facturas pendientes de pago...), se ejecutará al seleccionar la correspondiente tarea en una relación de opciones o *menú*.

En la medida en que crece la complejidad de las aplicaciones, éstas suelen obviar la presencia del traductor de lenguaje. Para optimizar su volumen (memoria necesaria) e incrementar su velocidad, estas aplicaciones suelen estar escritas en lenguaje máquina y, por consiguiente, no exigen la presencia del traductor de lenguaje de alto nivel. Dentro de la estructura software del ordenador, estas aplicaciones se apoyarán sobre el nivel ocupado por el sistema operativo.

Las aplicaciones más simples o que no exigen una elevada velocidad de ejecución, pueden introducirse directamente en lenguaje de alto nivel. En consecuencia, al tratarse de programas fuente, es necesario contar con el traductor adecuado que se ocupe de su conversión al lenguaje propio de la máquina.

¿COMO PROVEERSE DEL SOFTWARE DE APLICACION?

La mayor parte de las aplicaciones habituales de los ordenadores no suelen ser creaciones del propio usuario. La inversión en tiempo o en adquisición de conocimientos y utilidades para la programación, son algunos factores que mueven al usuario a optar por programas comerciales. A estos condicionantes, hay que añadir las contrapartidas que aportan muchos programas comerciales; por ejemplo, permiten una estandarización de las aplicaciones que facilita el intercambio de datos, conocimientos y soluciones entre los usuarios de una misma aplicación.

En definitiva, a la hora de dar contenido al tercer nivel del edificio de la programación, el usuario cuenta con tres alternativas básicas:

- Crear sus propios programas de aplicación.
- Encargar la confección de programas «a medida».
- Adquirir programas estandarizados.

La primera alternativa, resulta adecuada cuando se trata de confeccionar programas sencillos o destinados a aplicaciones cuya originalidad lo aconseje. No obstante, ésta no es una alternativa generalizable, puesto que, como ya se ha indicado, la complejidad de las aplicaciones puede convertir la tarea de programación en una empresa imposible, antieconómica o, en el mejor de los casos, puede exigir una excesiva inversión de tiempo.

Cuando las aplicaciones alcanzan una complejidad sustancial y deben ajustarse a unos criterios muy específicos, resulta conveniente encargar los programas «a medida». Cada vez son más las empresas dedicadas a la creación de software a medida. Su actividad primordial reside en el campo de las aplicaciones administrativas y de gestión especializada, para profesionales o empresas. Hay que tener en cuenta que la disparidad de situaciones que se plantean a la hora de gestionar la actividad de un almacén, o el control de una red comercial, reduce las posibilidades de estandarizar los respectivos programas y paquetes de aplicación.

CATEGORIAS DEL SOFTWARE DE APLICACION

Son muy diversos los criterios que pueden adoptarse para clasificar el software de aplicación destinado a ordenadores



A la hora de proveer al ordenador del adecuado software de aplicación, el usuario cuenta con tres alternativas básicas: crear sus propios programas, encargarlos «a medida» o adquirir programas estandarizados.



Los programas de juego constituyen el grupo más extenso y popular del software de aplicación.

SOFTWARE DE APLICACION

JUEGOS

CIENTIFICO/ TECNICO

EDUCACION

ADMINISTRACION

DEBE HABER

GESTION

personales. El primero y más genérico lleva a la distinción entre programas *profesionales,* auxiliares o *de utilidad y lúdicos.* Un segundo criterio de clasificación, menos genérico que el anterior, conduce a la distinción de cinco grupos fundamentales de programas y paquetes de aplicación:

- Juegos/Entretenimiento
- Educación
- Productividad y gestión
- Científico-técnicos
- Contabilidad y administración

Cada uno de los anteriores grupos admite, a su vez, una clasificación específica por categorías de funcionalidad.

¿Para qué sirve un ordenador personal? La respuesta cabe encontrarla en el ingente repertorio de programas capaces de resolver las tareas más dispares.

JUEGOS

Este grupo engloba a los programas más populares en el terreno de los ordenadores personales que ocupan el estrato inferior (ordenadores de bolsillo y familiares); si bien, también abundan los programas lúdicos destinados a ordenadores personales más evolucionados, por ejemplo, para equipos profesionales como los modelos de la firma Apple o para el IBM-PC. Dentro de este apartado, cabe diferenciar entre varias categorías de juegos: juegos denominados de *«arcade»,* entre los que destaca el popular «Space Invaders» o el más reciente «Zaxxon»; juegos de *aventuras,* de *acción,* de *estrategia* y de *simulación,* además de los *clásicos* y no por ello menos interesantes, como son el ajedrez, damas, Monopoly, Othello o Intelect.

EDUCACION

Dentro de este apartado caben desde programas para el estudio de matemáticas, geografía o historia, hasta complejos pro-



Desde simples programas para el estudio de los conceptos matemáticos, hasta complejos paquetes para «enseñanza asistida por ordenador». Un vasto marco de aplicaciones didácticas cuyo eco es notable en el mundo del software de aplicación.



Las herramientas creadas para la ayuda a la gestión (hojas electrónicas, gestores de bases de datos, tratamientos de textos...), han convertido al ordenador personal en un inapreciable colaborador del profesional y del gestor de empresa.

gramas de *enseñanza asistida por ordenador* para el aprendizaje de idiomas o de lenguajes de programación de alto nivel (BASIC, PILOT, LOGO...).

PRODUCTIVIDAD Y GESTION
El mayor segmento del mercado de software estandarizado lo ocupa esta categoría del software de aplicación. Tratamiento de textos, hojas electrónicas, gestores de bases de datos, generadores de programas, paquetes para la generación de gráficos o para el establecimiento de comunicaciones entre equipos. Sin olvidar a los nuevos *paquetes integrados* multifuncionales (Lotus 1.2.3, Open Access, Symphony...). Todo un repertorio de herramientas habituales que permiten al ordenador personal acometer con éxito aplicaciones de gestión y productividad.

CIENTIFICO-TECNICOS
Los profesionales cuentan con un amplio catálogo de aplicaciones, creadas para apoyar casi cualquier tipo de actividad científica con la colaboración del ordenador personal. Programas destinados a profesionales de la medicina (gestión de datos de los pacientes y archivo de diagnósticos), aplicaciones para arquitectos, abogados, ingenieros...

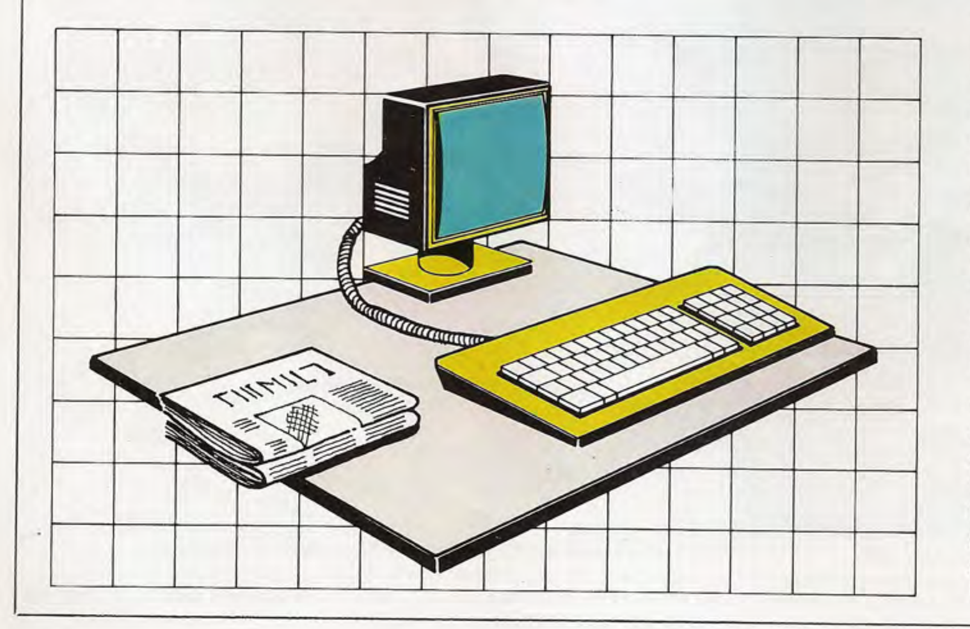
CONTABILIDAD Y ADMINISTRACION
Sin lugar a dudas, éste es uno de los grupos de aplicaciones que más ha potenciado la difusión del ordenador personal en el ámbito de la empresa: gestión de la contabilidad, nóminas, almacén, control de pedidos, libro de vencimientos...

El ordenador en actividad

La expansión de los ordenadores y su vertiginosa entrada en todos los campos de actividad, es paralela a la andadura del software de aplicación. Los cada vez más potentes y perfeccionados paquetes de aplicación amplían el área de actuación del ordenador, convirtiéndolo en una herramienta insoslayable en las tareas más dispares.

Uno de tantos ejemplos lo obtenemos en el trabajo periodístico. Máquinas de escribir, lapiceros, gomas de borrar, ingentes archivos de datos, referencias y citas, ceden su paso al ordenador.





Un simple ordenador personal dotado de su correspondiente sistema operativo, completado con un paquete estandarizado de tratamiento de textos y una impresora, constituyen el entorno de trabajo del periodista o del escritor actual.

La goma de borrar, la ineludible tachadura e incluso la imagen de la papelera repleta de originales descartados, se diluye ante la potencia de un tratamiento de textos.

La edición del texto en la pantalla está apoyada por un amplio repertorio de funciones: borrado de caracteres, palabras, líneas y párrafos; inserción de nuevo texto dentro de un original en edición; redistribución de párrafos y apartados; inclusión automática de textos anteriores...

Funciones que permiten una cómoda edición y puesta a punto de los originales que se almacenarán en un simple disco flexible. Y que, además, garantizan una impresión final impecable, en la que es posible definir con detalle cualquier precisión relativa al formato de escritura y a la distribución del texto sobre el papel.

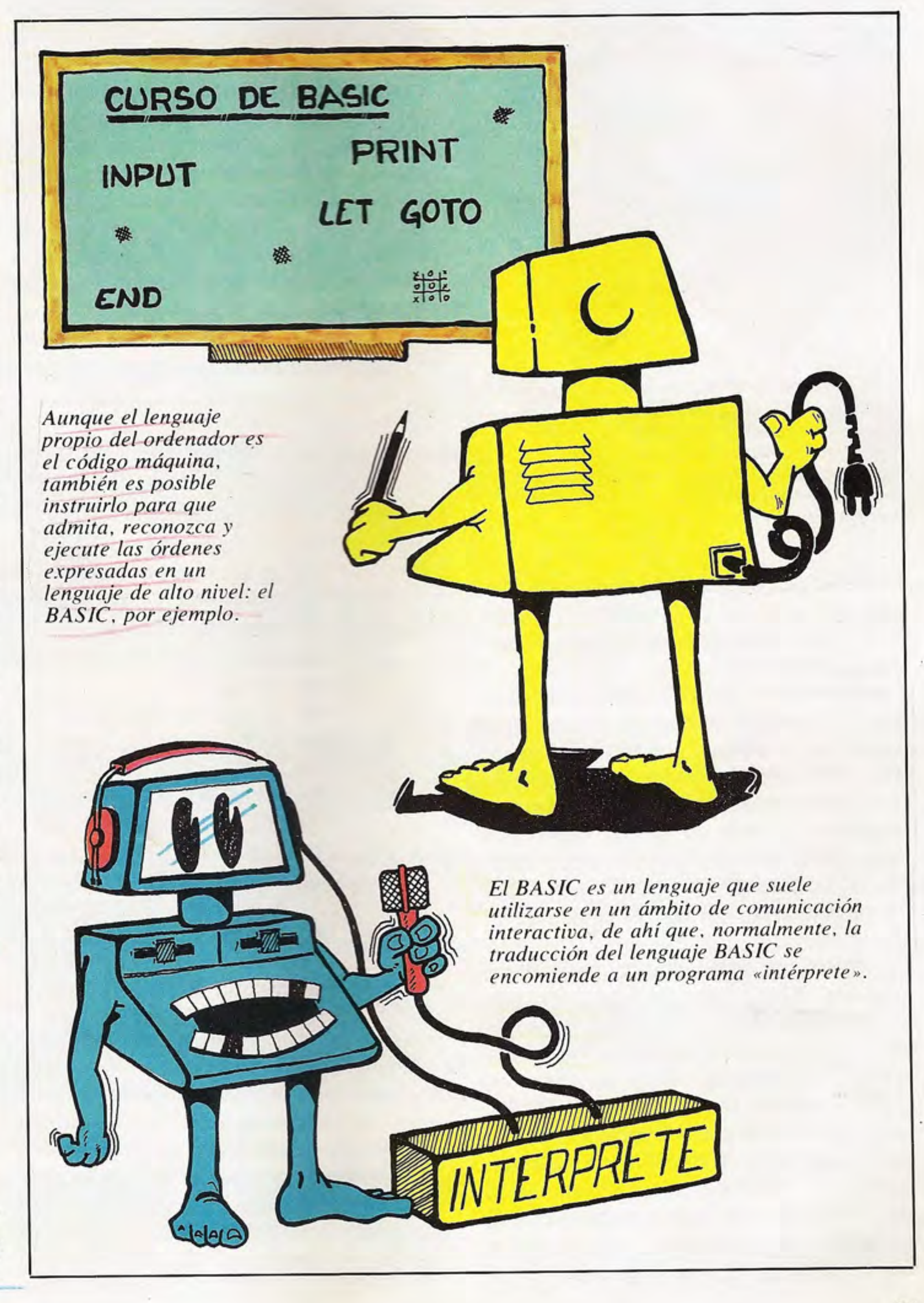
Educando a la máquina

Lenguaje, programas, instrucciones y comandos

TODO ordenador personal está regido por un minúsculo cerebro, el *microprocesador,* cuyo aspecto es el de un rectángulo de plástico negro, flanqueado por un gran número de patillas metálicas. En su interior se alojan una gran cantidad de dispositivos electrónicos, integrados hasta el punto de ocupar un volumen microscópico. En su conjunto, constituyen un peculiar circuito lógico, capaz de gobernar su propia actividad y la del resto de los circuitos que componen el ordenador.

Semejante cerebro integrado puede realizar casi cualquier tipo de operación con los datos que reciba; no obstante, tiene una seria limitación: alguien ha de comunicarle lo que debe hacer, es preciso instruirlo con toda suerte de detalles.

Pero..., ¿cómo hay que hablar con el ordenador? Debido a la naturaleza de su cerebro, un circuito electrónico lógico, resulta obvio que podremos comunicarnos directamente a base de sucesiones de unos y ceros: empleando el lenguaje máquina, ni más ni menos. Cualquier frase formulada en este lenguaje, tiene el aspecto externo de una cadena de unos y ceros. Para un observador ajeno a las artes de programación, su significado es críptico; sin embargo, para el ordenador representan las únicas órdenes e instrucciones que puede entender y ejecutar.



Aunque el lenguaje propio del ordenador es el código máquina, también es posible instruirlo para que admita, reconozca y ejecute las órdenes expresadas en un lenguaje de alto nivel: el BASIC, por ejemplo.

El BASIC es un lenguaje que suele utilizarse en un ámbito de comunicación interactiva, de ahí que, normalmente, la traducción del lenguaje BASIC se encomiende a un programa «intérprete».

EL ARTE DE DIALOGAR EN BASIC

Si el ordenador sólo es capaz de entender los arcanos del cero y el uno, ¿cómo es posible que los ordenadores entiendan el

repertorio de frases, muy parecidas al lenguaje ordinario, que pueden construirse con el vocabulario de un lenguaje evolucionado como el BASIC?
Nada más sencillo... ¿Para qué están los traductores? Dada la enorme capacidad de trabajo del ordenador, también podrá encargarse de convertir nuestras frases en BASIC a las secuencias de ceros y unos de su propio lenguaje interno.
La máquina aguarda a que se la instruya, y no hay óbice que impida que el primer programa que reciba sea, precisamente, un traductor. Como ya sabemos, los traductores de lenguajes pueden ser de dos tipos: *intérpretes* o *compiladores.*

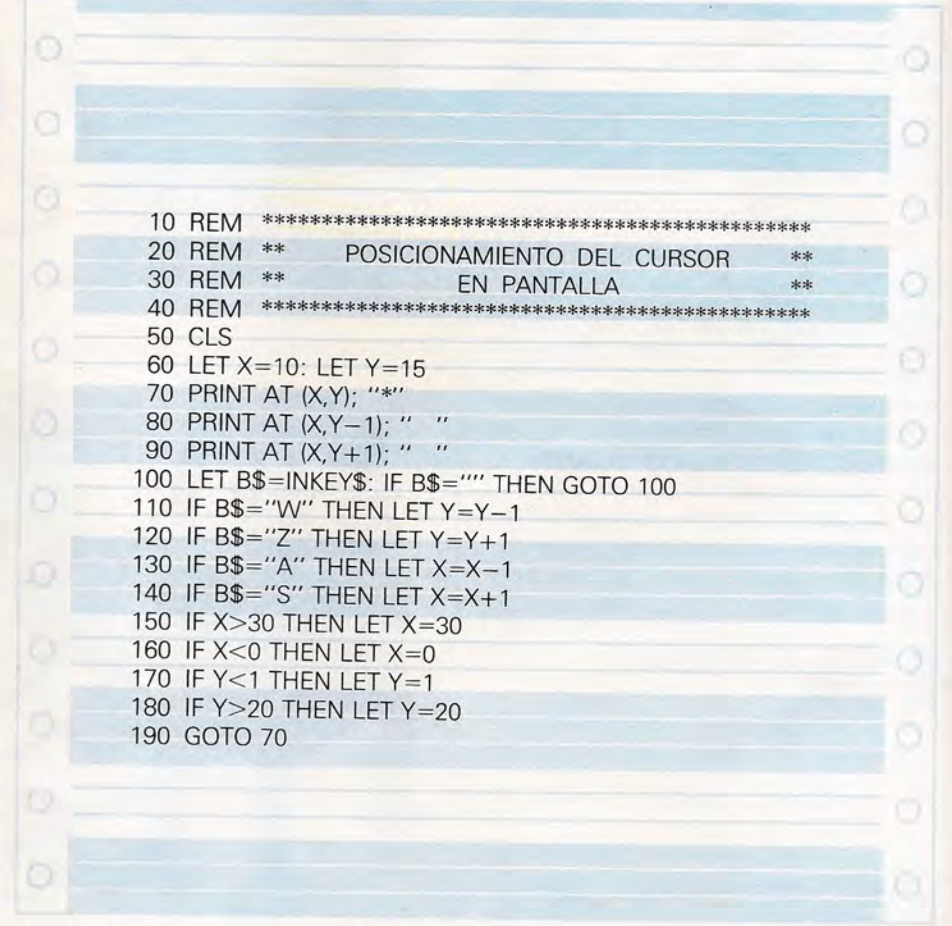


Una instrucción es una palabra o frase que educa a la máquina para que ejecute una operación o tarea específica. Agrupando un determinado número de frases es posible construir un programa, capaz de instruir a la máquina para que realice una actividad completa.

El BASIC es un lenguaje que suele utilizarse en un ámbito interactivo y, por ello, se emplea habitualmente a modo de lenguaje interpretado.
Cabe recordar que la actuación del programa traductor tendrá un total paralelismo con la realidad cotidiana. Por ejemplo, si un científico extranjero quiere dar una conferencia y ni él sabe castellano, ni el público conoce su idioma, caben dos soluciones: que un intérprete vaya traduciendo su disertación, paso a paso, o que se reparta una hoja con el contenido ya traducido.
En efecto, el ordenador no entiende el lenguaje BASIC directamente, sino que antes de asimilar su contenido, debe traducirlo a su propio código (el lenguaje máquina). Tal como ya hemos señalado, las comunicaciones con la máquina en BASIC son, con frecuencia, interactivas; de ahí que el traductor de BASIC a lenguaje máquina sea, habitualmente, un intérprete de BASIC. No obstante, cuando el BASIC se utilice para programar tareas complejas, cuya posterior ejecución debe ser rápida, es obvio que resultará más adecuado sustituir el intérprete por un compilador BASIC.
La ejecución de un programa compilado (traducido en bloque) es bastante más rápida que la ejecución de un programa canalizado a través de un intérprete. Volviendo al ejemplo anterior, la conferencia se prolongará por más tiempo si un intérprete debe traducir frase a frase la disertación del conferenciante; por el contrario, si el texto de la conferencia se traduce de una sola vez (tarea del compilador) y se entrega traducido a los presentes, la duración de la conferencia será bastante más breve.
A la hora de establecer un primer contacto con el mundo de la programación en BASIC, o incluso acometer la confección de programas, el intérprete BASIC se revela como un eficaz auxiliar. Entre sus funciones están las de detectar errores y comunicarlos al usuario, además de facilitar las tareas de corrección y depuración de los programas.

INSTRUCCIONES Y PROGRAMAS

Al igual que cualquier lenguaje humano, el BASIC se rige por un conjunto de normas semánticas y reglas sintácticas cuya puesta en práctica permite construir mensajes organizados. La base del lenguaje se encuentra en su vocabulario o repertorio de palabras a través de las cuales es posible expresar cualquier acción programable. El vocabulario del BASIC está constituido por palabras del idioma inglés. Un vocabulario bastante más simple y limitado que el de cualquier lenguaje humano, pero suficiente para construir las órdenes destinadas a la máquina. Tal como ocurre en los diálogos humanos, la unidad de comunicación se encuentra en la frase, o con-



```
10 REM ********************************************
20 REM **      POSICIONAMIENTO DEL CURSOR        **
30 REM **             EN PANTALLA                **
40 REM ********************************************
50 CLS
60 LET X=10: LET Y=15
70 PRINT AT (X,Y); "*"
80 PRINT AT (X,Y-1); "  "
90 PRINT AT (X,Y+1); "  "
100 LET B$=INKEY$: IF B$="" THEN GOTO 100
110 IF B$="W" THEN LET Y=Y-1
120 IF B$="Z" THEN LET Y=Y+1
130 IF B$="A" THEN LET X=X-1
140 IF B$="S" THEN LET X=X+1
150 IF X>30 THEN LET X=30
160 IF X<0 THEN LET X=0
170 IF Y<1 THEN LET Y=1
180 IF Y>20 THEN LET Y=20
190 GOTO 70
```

Aspecto de un programa en lenguaje BASIC.

junto de palabras que expresan una determinada acción. En el caso del BASIC, la unidad de comunicación se denomina *instrucción.*

El objeto de una instrucción es «educar» al ordenador para que éste realice una tarea específica.

Agrupando un cierto número de frases, es posible expresar una actividad completa con todos sus matices. Lo mismo ocurre en el ámbito de los lenguajes de programación al asociar un determinado número de instrucciones destinadas a la máquina. El conjunto organizado de instrucciones que definen una tarea completa recibe el nombre de *programa.*

La estructura de las frases del lenguaje común se mantiene, con ligeras salvedades, en las frases o instrucciones que conforman un programa en lenguaje BASIC. Un verbo es quien define la acción a realizar por el sujeto; análogamente, un *comando* es el que indica la acción que debe realizar el ordenador con los datos.

En una instrucción BASIC intervienen dos zonas:

- Comando
- Argumento

El comando (semejante al verbo de una frase convencional) expresa la acción. Éste puede coincidir con una simple orden, desprovista de argumento, dirigida a la máquina. Por ejemplo:

STOP: Detener la secuencia de ejecución.
NEW: Borrar por completo el programa en curso.
RUN: Ejecutar el programa.

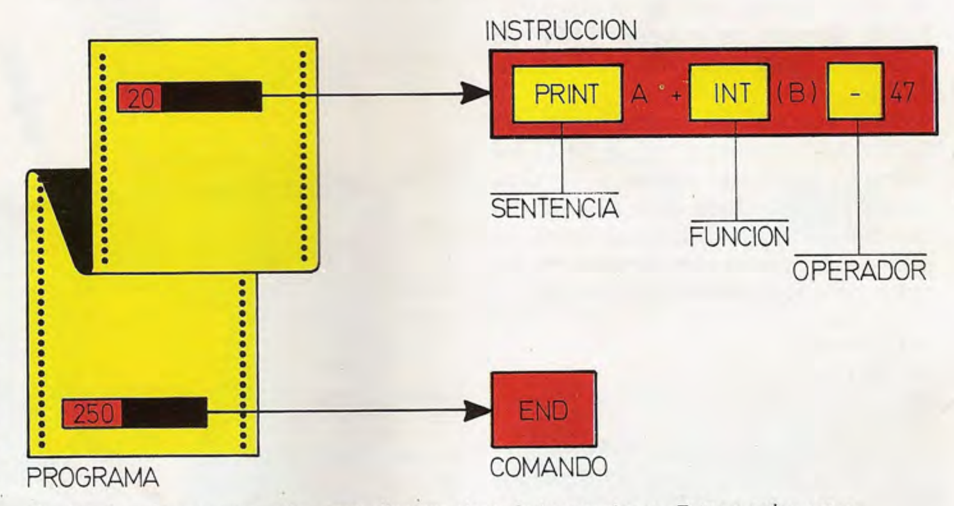
Cuando la acción afecta a un dato, a un grupo de datos, o a un dispositivo asociado al ordenador, el comando se acompaña de una segunda zona denominada *argumento.* Si la instrucción incluye ambas zonas, el comando suele recibir el nombre de *sentencia.*

```
LET A=20
PRINT 430
GOTO 25
```

Todas estas instrucciones incluyen una sentencia (comando) seguida por un argumento que aporta el dato o datos implicados en la ejecución de la orden. Por ejemplo, la primera instrucción apuntada (LET A=20) insta a la máquina para que asigne el valor 20 a la variable A. La segunda (PRINT 430) comunica al ordenador que lleve a la pantalla el dato 430; mientras que GOTO 25 ordena un salto a la instrucción número 25.



El ordenador doméstico, rodeado de un entorno mínimo, constituye un perfecto trampolín para entrar en el mundo de la programación en BASIC.



Dentro de un programa caben muy diversos tipos de instrucciones. Estas pueden estar constituidas por un comando asociado a un argumento (dato o datos relacionados por funciones y/o operadores), o por un comando aislado que expresa una orden.

Dentro de la zona de argumento caben muy diversos tipos de datos y con distintas expresiones: valores numéricos, palabras o grupos de caracteres alfanuméricos, variables, referencias a dispositivos periféricos... Habitualmente, el argumento contiene varios datos relacionados entre sí por *operadores* (suma, resta, multiplicación, igualdad...), o afectados por *funciones,* ya sean matemáticas o de cualquier otro tipo autorizado por el lenguaje BASIC.

INSTRUCCIONES DIRECTAS E INDIRECTAS

En un diálogo caben tanto frases sueltas, que dan pie a una respuesta inmediata,

como mensajes que agrupan a un determinado número de frases. Esta es una realidad aplicable también al lenguaje BASIC. El usuario puede dirigirse al ordenador con una simple instrucción y aguardar su respuesta. Si bien, también puede comunicarse recurriendo a una secuencia de instrucciones o programa.

En ambos casos las instrucciones son las mismas, aunque se emplean de distinta forma. Cuando la instrucción se utiliza de forma independiente, para que la máquina la ejecute y curse una respuesta inmediata, recibe el nombre de *instrucción directa*. Por el contrario, si ésta forma parte de un programa y se encuentra precedida por un número de orden que denota su situación dentro del mismo, se estará formulando como *instrucción indirecta*.

Al accionar la tecla RETURN tras una instrucción en modo directo, el ordenador la asimilará y ejecutará sin mayor dilación. Por ejemplo, si se introduce la instrucción LET A=20, al pulsar la tecla RETURN el ordenador asignará de inmediato el valor 20 a la variable A.

De forma análoga, tras dar la orden RETURN, después de introducir PRINT 35 aparecerá en la pantalla el dato indicado (35).

Ambas instrucciones pueden agruparse dentro de un programa, lo que equivale a utilizarlas en modo indirecto. En este caso, cada instrucción debe estar precedida por un número (número de línea) que servirá al ordenador para discernir en qué orden debe ejecutarlas. Por ejemplo:

```
10 LET A=250
20 PRINT A
■
```

Las instrucciones utilizadas en modo indirecto, ya no son ejecutadas por el ordenador al accionar la tecla RETURN que pone fin a cada línea. Estas pasan a constituir un programa que será ejecutado en bloque al introducir la orden al efecto: RUN.

```
10 LET A=250
20 PRINT A
RUN

250
■
```

En el ejemplo, al recibir la orden RUN, el ordenador ha ejecutado ambas instrucciones en el orden indicado por el número de línea que las acompaña. Al ejecutar la instrucción 10 (LET A=250) se asigna a la variable A el valor 250. Acto seguido, se ejecuta la instrucción 20 que ordena imprimir el valor de la variable A.

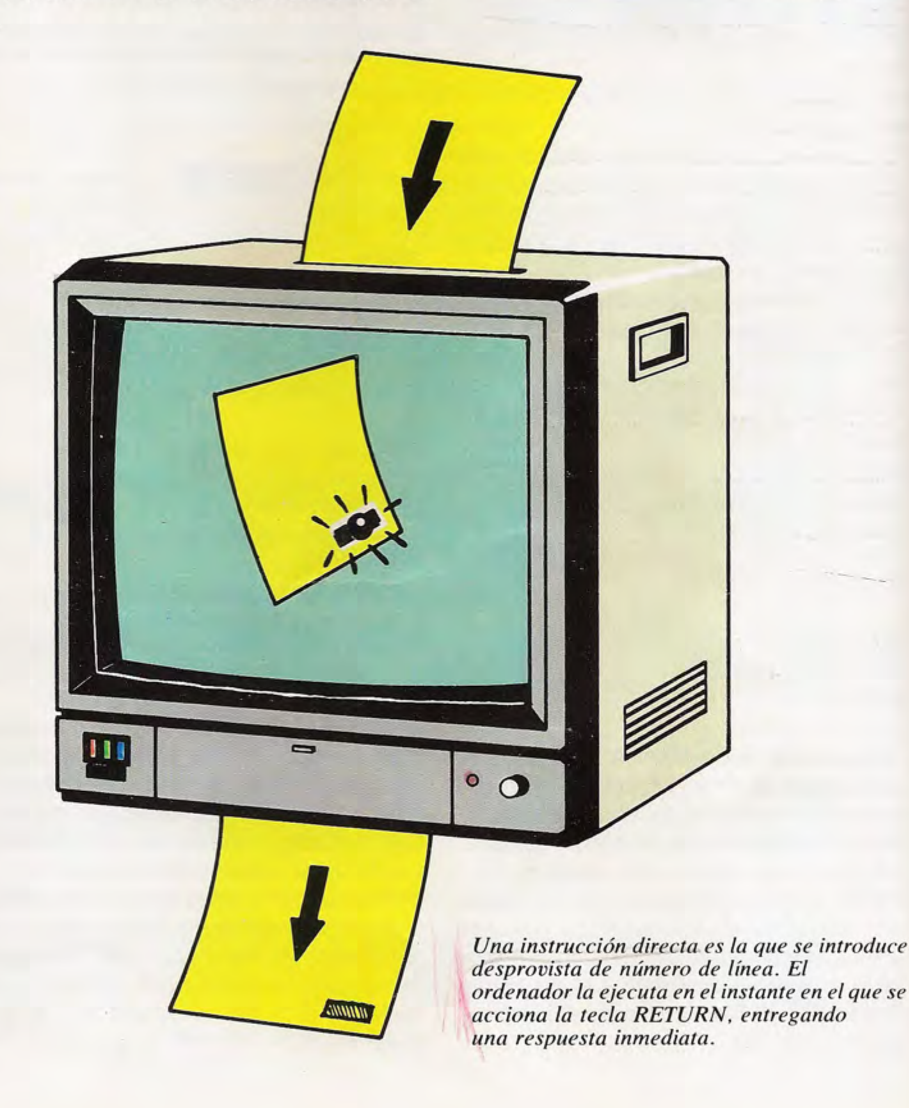
ASPECTO DE UN PROGRAMA BASIC

Como ya hemos visto, un programa es una secuencia ordenada de instrucciones que definen una tarea completa. Las instrucciones se distribuyen en líneas sucesivas, cada una de ellas precedida por un número entero que señala el orden en el que será ejecutada por el ordenador.

Los números de línea deben seguir un orden creciente. De esta forma, la ejecución de la secuencia de instrucciones hará que el ordenador realice las sucesivas operaciones que conducirán al resultado final.

Un programa redactado en lenguaje BASIC presenta una estructura semejante a la que sigue:

```
10 INSTRUCCION 1
20 INSTRUCCION 2
30 INSTRUCCION 3
40 INSTRUCCION 4: INSTRUCCION 5
50 INSTRUCCION 6
55 END
■
```



Una instrucción directa es la que se introduce desprovista de número de línea. El ordenador la ejecuta en el instante en el que se acciona la tecla RETURN, entregando una respuesta inmediata.

El número de línea instruye al ordenador sobre el orden en el que debe ejecutar el programa. Cabe observar que, excepto en la última línea de instrucción, se han numerado las líneas a intervalos de 10. ¿Por qué..? En la práctica, el primer intento de escritura de un programa muy raramente se ve coronado por el éxito; lo habitual es que queden en el olvido algunas instrucciones. Numerando las líneas con un cierto intervalo, queda abierta la posibilidad de añadir nuevas instrucciones en el lugar oportuno, utilizando números de línea intermedios. Recordemos que el ordenador ejecutará el programa atendiendo estrictamente a un orden numérico creciente; así pues, si hemos olvidado una instrucción intermedia entre las instrucciones 2 y 3 no tenemos por qué reescribir el programa completo, sino que bastará simplemente con escribir al final del programa la citada instrucción con el número de línea «a propósito», por ejemplo: 24. El intérprete BASIC se ocupará de que las instrucciones añadidas posteriormente se sitúen en el lugar que les corresponda dentro del programa, atendiendo a su número de línea.

A la hora de escribir un programa, la orden RETURN tiene la misma importancia que «el retroceso de carro» en el caso de una máquina de escribir. Al dar por concluida la escritura de una línea, resulta imperativo dar la orden RETURN. El intérprete BASIC responderá de inmediato, mostrando en la pantalla el cursor o «indicador de presencia»; éste señalará la posición a partir de la que se escribirá la próxima instrucción.

Algunos intérpretes BASIC permiten la escritura de dos o más instrucciones en una misma línea de programa. Las diversas instrucciones presentes en la misma línea se separan por medio de un carácter denominado «separador de instrucción». En el ejemplo propuesto (línea 40) las instrucciones 4 y 5 se separan por medio del signo ":".

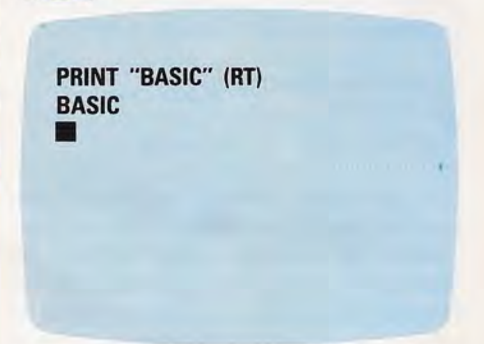
La necesidad de instruir al ordenador con toda suerte de precisiones, llega hasta el punto que es preciso señalar el final del programa. De ahí que la última línea del programa esté ocupada por una instrucción BASIC especializada en tal menester; ésta suele coincidir con la orden END.

Las instrucciones BASIC recuerdan a las fórmulas matemáticas. Una semejanza que apunta la utilidad del BASIC para resolver problemas científicos. A pesar de ello, el campo de acción de este lenguaje de alto nivel cubre todo tipo de disciplinas, desde los juegos y el aprendizaje hasta las tareas de gestión. Su naturaleza de lenguaje interactivo —el usuario recibe una respuesta instantánea a sus comunicaciones—, ha contribuido a que el BASIC sea el lenguaje más utilizado y popular.

EL COMANDO PRINT

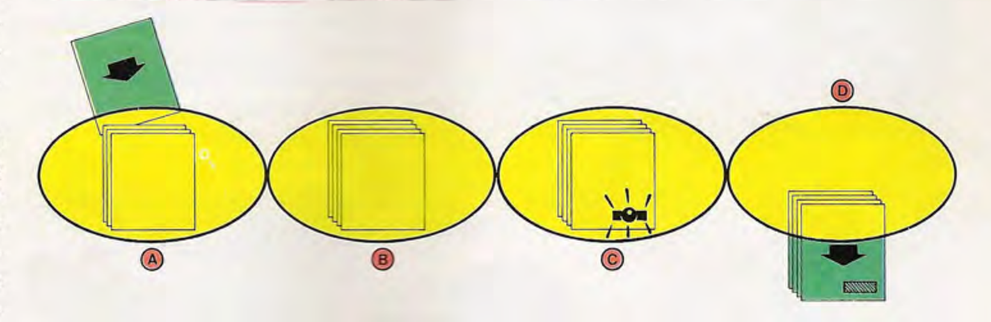
La primera impresión de que el ordenador es una máquina con capacidad práctica, la obtenemos al observar la pantalla repleta de mensajes y dibujos. De ahí que uno de los primeros objetivos de todo aquel que

por concluida la instrucción con la orden RETURN.



(RT): Acción sobre la tecla RETURN.

Desde luego, la mayor parte de las veces, la instrucción PRINT se utiliza en modo indirecto, formando parte de un programa. En este caso, y tal como puede observarse en el programa que sigue, la ejecución tendrá lugar al comunicarle al ordenador la orden RUN:



Las instrucciones indirectas, o precedidas por un número de línea, se integran dentro de un programa (A y B) que sólo será ejecutado por el ordenador (C y D) al recibir la orden RUN.

empieza a programar sea, precisamente, escribir algo en la pantalla.

El lenguaje BASIC dispone de varios comandos especializados en esta función; entre ellos, el más importante es PRINT. Utilizándolo, en modo directo o indirecto, pueden llevarse a la pantalla los números, letras o palabras que desee el usuario.

El formato más simple de una instrucción PRINT es el que incluye el mencionado comando, seguido por un texto encerrado entre comillas. Por ejemplo:

```
PRINT "BASIC"
PRINT "M-6727"
PRINT "LENGUAJE DE PROGRAMACION"
```

Al utilizar la instrucción PRINT en modo directo (sin número de línea), el ordenador llevará a la pantalla el texto encerrado entre comillas, en el instante en el que se dé

```
10 PRINT "SOFT:"
20 PRINT "CURSO PRACTICO"
30 PRINT "DE PROGRAMACION"
40 END
RUN

SOFT:
CURSO PRACTICO
DE PROGRAMACION
■
```

Cuando el dato a imprimir es un número, se omiten las comillas. Por ejemplo:

```
10 PRINT 15
20 PRINT 356.2
```

Lo mismo ocurre cuando el argumento de la instrucción PRINT está constituido por nombres de variables, ya sean numéricas

(A, B, C...) o de cadena de caracteres (A$, B$, C$...). En tal caso, la instrucción PRINT imprimirá los valores que estén asignados a las variables en cuestión:

```
10 LET A$="EL NUMERO ES:"
20 LET N=8
30 PRINT A$;N
```

Las anteriores instrucciones constituyen un verdadero programa, al que sólo hay que añadir la instrucción final END. Su ejecución, ordenada por medio del comando RUN, ilustra cuál es el efecto de los nombres de variables dispuestos en el argumento de PRINT:



PRINT

Imprime en la pantalla los mensajes o el valor de las expresiones que aparecen en la zona de argumento.

Formato: (Número de línea) PRINT <*expresión 1*> {[,] [;]} <*expresión 2*>...

Ejemplos: PRINT 25
10 PRINT "SOFT"
35 PRINT A,C$,"CURSO DE BASIC"

Notación utilizada en el formato:
< >: Los textos y expresiones encerradas por los símbolos "menor" o "mayor" son aportación del usuario.
[ ]: Los elementos encerrados entre corchetes son opcionales.
{ }: Las llaves delimitan a los elementos alternativos.

```
10 LET A$="EL NUMERO ES:"
20 LET N=8
30 PRINT A$
40 PRINT N
50 END
RUN

EL NUMERO ES:
8
■
```

En efecto, el ordenador ha llevado a la pantalla el contenido de ambas variables: una cadena de caracteres en el caso de A$ y un valor numérico en el de la variable N.
No terminan aquí las posibilidades de esta versátil instrucción. El argumento de PRINT puede también estar constituido por una expresión alfanumérica o matemática; ésta se verá resuelta por el ordenador antes de proceder a la presentación del resultado en pantalla.

```
10 LET A=5
20 PRINT "SUMA"
30 PRINT 10+20
40 PRINT 15-A+2
50 END
RUN

SUMA
30
12
■
```



SEPARADORES EN EL ARGUMENTO DE PRINT

Una misma instrucción PRINT puede utilizarse para trasladar a la pantalla diversos datos. Estos deben incluirse dentro de la zona de argumento, convenientemente separados. Habitualmente, los signos que admite cualquier intérprete BASIC para separar a los distintos datos que acompañan al comando PRINT son la coma (,) y el punto y coma (;).
Al utilizar la coma como signo de separación, los datos aparecerán en la pantalla distribuidos en campos de una determinada longitud. Por el contrario, cuando el



¿Cómo hay que hablar de un ordenador? ¿Es posible establecer una comunicación sin recurrir al código máquina?.. El BASIC constituye un claro ejemplo de lenguaje de alto nivel capaz de diluir la distancia entre el usuario y la máquina.

TABLA DE CONVERSION

ORDENADOR	PRINT						
	PRINT	PRINT,	PRINT;	PRINT "<Mens.>"	PRINT X	PRINT <exp.>	PRINT <carácter de control>
APPLE II	PRINT	PRINT,	PRINT;	PRINT "<Mens.>"	PRINT X	PRINT <exp.>	
APRICOT	PRINT	PRINT,	PRINT;	PRINT "<Mens.>"	PRINT X	PRINT <exp.>	
ATARI	PRINT	PRINT,	PRINT;	PRINT "<Mens.>"	PRINT X	PRINT <exp.>	PRINT <carácter de control>
CBM 64	PRINT	PRINT,	PRINT;	PRINT "<Mens.>"	PRINT X	PRINT <exp.>	PRINT <carácter de control>
DRAGON	PRINT	PRINT,	PRINT;	PRINT "<Mens.>"	PRINT X	PRINT <exp.>	
EQUIPOS MSX	PRINT	PRINT,	PRINT;	PRINT "<Mens.>"	PRINT X	PRINT <exp.>	
HP-150	PRINT	PRINT,	PRINT[;]	PRINT "<Mens.>"	PRINT X	PRINT <exp.>	
IBM PC	PRINT	PRINT,	PRINT;	PRINT "<Mens.>"	PRINT X	PRINT <exp.>	
MPF	PRINT	PRINT,	PRINT;	PRINT "<Mens.>"	PRINT X	PRINT <exp.>	
NCR DN-V	PRINT	PRINT,	PRINT;	PRINT "<Mens.>"	PRINT X	PRINT <exp.>	
NEW BRAIN	PRINT	PRINT,	PRINT;	PRINT "<Mens.>"	PRINT X	PRINT <exp.>	
ORIC	PRINT	PRINT,	PRINT;	PRINT "<Mens.>"	PRINT X	PRINT <exp.>	
QL	PRINT	PRINT,	PRINT;	PRINT "<Mens.>"	PRINT X	PRINT <exp.>	
SHARP MZ-700	PRINT	PRINT,	PRINT;	PRINT "<Mens.>"	PRINT X	PRINT <exp.>	PRINT <carácter de control>
SPECTRAVIDEO	PRINT	PRINT,	PRINT;	PRINT "<Mens.>"	PRINT X	PRINT <exp.>	
SPECTRUM	PRINT	PRINT,	PRINT;	PRINT "<Mens.>"	PRINT X	PRINT <exp.>	

"<Mens.>": Mensaje o texto, entre comillas. <Exp.>: Expresión con dato o datos numéricos o alfanuméricos.

FORMULACIONES DEL COMANDO PRINT

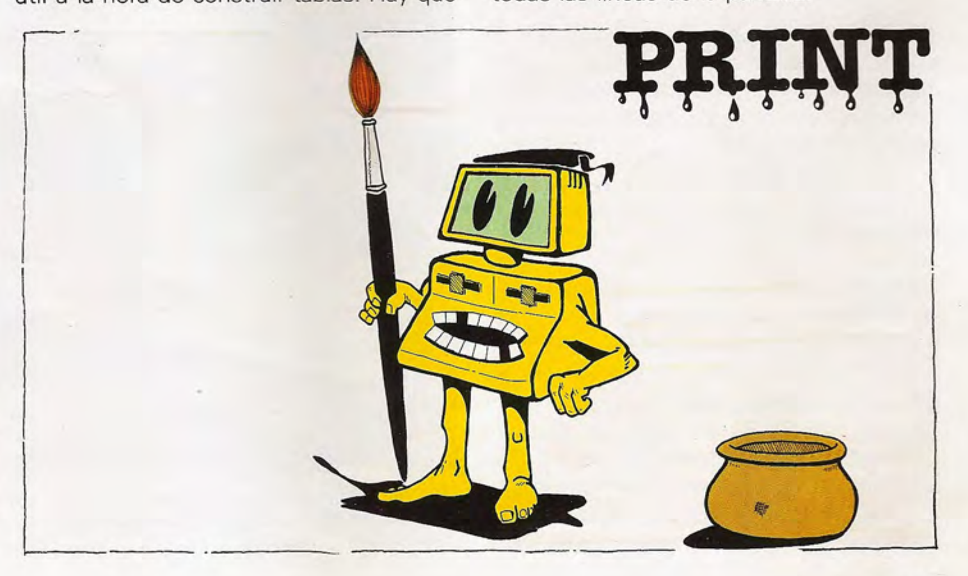
PRINT: Salto a la próxima línea dejando una en blanco. PRINT,: Coloca el cursor unos espacios más adelante. PRINT;: Deja el cursor en el punto de impresión, sin desplazarlo. PRINT "<mensajes>": Escritura en pantalla del mensaje encerrado entre comillas. PRINT X: Escritura en pantalla del valor asignado a la variable X. PRINT (expresión): Imprime el valor de la expresión. PRINT <carácter de control>: Ejecuta la acción ordenada por el carácter de control.

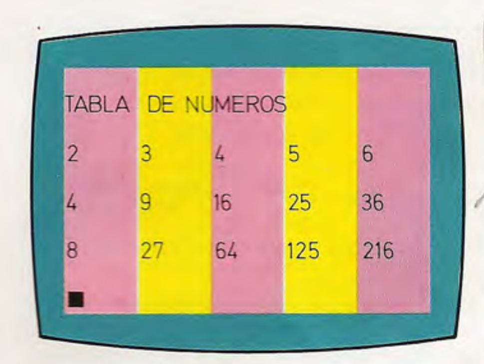
separador es el punto y coma, los datos se imprimirán uno inmediatamente a continuación de otro. Por ejemplo:

```
10 PRINT "DOS:",2
20 PRINT "TRES:";3
30 END
RUN

DOS:        2
TRES:3
■
```

Al encontrar la coma, el ordenador escribirá ambos datos dejando un determinado espacio en blanco; éste será mayor o menor dependiendo de las características de cada intérprete BASIC. La distribución de campos o zonas de impresión dentro de una misma línea, originada por la presencia de este signo separador, resulta muy útil a la hora de construir tablas. Hay que tener en cuenta al respecto, que la distribución de los campos se mantiene en todas las líneas de la pantalla.






```
10 PRINT "TABLA DE NUMEROS"
20 PRINT 2,3,4,5,6
30 PRINT 4,9,16,25,36
40 PRINT 8,27,64,125,216
50 END
```

Tal como se observa al ejecutar el programa adjunto, al emplear la coma (,) como separador de los datos de PRINT, la pantalla se divide en un determinado número de campos o zonas de impresión (el número de campos y su longitud dependen de cada intérprete BASIC).

```
RUN

NUMEROS

1        2        3
456789
■
```

Glosario

LENGUAJE MAQUINA: Repertorio de instrucciones en código binario directamente interpretables por el ordenador.

PROGRAMA FUENTE: Programa escrito por el usuario en un lenguaje de programación distinto del código máquina.

PROGRAMA OBJETO: Programa resultante de la traducción del programa fuente a código máquina.

INSTRUCCION: Orden elemental que comunica una acción al ordenador. La instrucción suele constar de dos zonas: comando y argumento. El comando define la acción, mientras que el argumento aporta los datos sobre los que hay que realizarla.

RETORNO DE CARRO: Orden de salto al principio de la próxima línea. Esta acción habitual en las máquinas de escribir es trasladable al ámbito de la programación BASIC. En este caso, la función de retorno está asociada a una acción sobre la tecla RETURN.

CADENA DE CARACTERES: Secuencia de caracteres alfanuméricos (letras, cifras y símbolos) que conforman una unidad de información tratable directamente por el ordenador.

EJECUCION: Puesta en práctica de las órdenes expresadas por medio de las instrucciones de un programa.

ARCHIVO: Conjunto organizado de datos residente en una unidad para el almacenamiento de información asociada al ordenador.

MODO COMANDO: Modo de funcionamiento interactivo, en el cual se pueden introducir comandos y ejecutarlos a continuación, o bien escribir un programa.

Los mencionados separadores («coma» y «punto y coma») también pueden utilizarse, con el mismo efecto, al final de la instrucción PRINT. En ambos casos, el cursor que señala el punto de impresión pasará a ocupar la posición que corresponda al separador utilizado antes de imprimir el argumento de la próxima instrucción PRINT. Veamos un programa claramente ilustrativo de ambas posibilidades:

```
10 PRINT "NUMEROS"
15 PRINT
20 PRINT 1,2,
30 PRINT 3
40 PRINT 4;5;6;
50 PRINT 7;8;9
60 END
```

Y éste es el resultado de su ejecución:

En efecto, la presencia de la coma al final de la instrucción 20 desactiva el salto a la próxima línea de pantalla, de tal forma que el dato 3 aportado por la siguiente instrucción PRINT, aparece en el próximo campo de la misma línea.
El punto y coma que cierra la instrucción 40 también inhibe el salto a la próxima línea, si bien, en este caso, el próximo dato (7) se visualizará adosado al último dato que figura en el argumento de la instrucción 40.
La instrucción 15 del ejemplo anterior ilustra otra de las peculiaridades del PRINT. Una instrucción PRINT puede estar constituida exclusivamente por el comando, desprovisto de argumento alguno. Su ejecución hará que aparezca en la pantalla una línea en blanco, pasando el cursor a ocupar la próxima línea de impresión.



Son muy diversos los dialectos BASIC que coexisten en la actualidad. La mayor parte de los fabricantes incluyen en sus equipos un intérprete BASIC con ciertas peculiaridades en su vocabulario y sintaxis.

Logo (1)

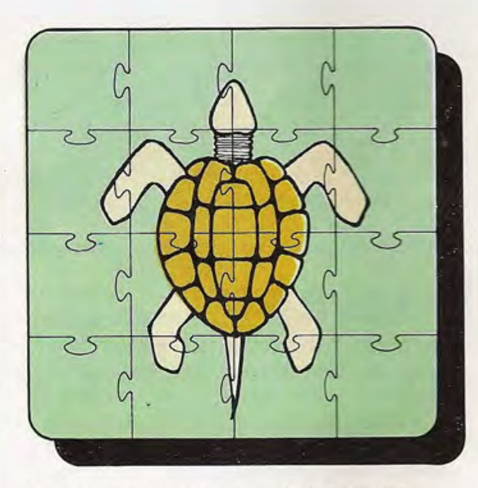
El lenguaje de la escuela

EL lenguaje LOGO fue desarrollado por Seymour Papert en el Laboratorio de Inteligencia Artificial del M.I.T. (Massachusetts Institute of Technology). Papert y su grupo buscaban un sistema que resultara sencillo para la enseñanza del uso de los ordenadores. Tras varios meses de investigación, surgió el LOGO: un lenguaje moderno nacido bajo la inspiración del LISP. Este último es un lenguaje evolucionado, orientado a aplicaciones de inteligencia artificial, que en 1959 vio la luz desde los propios laboratorios del M.I.T.

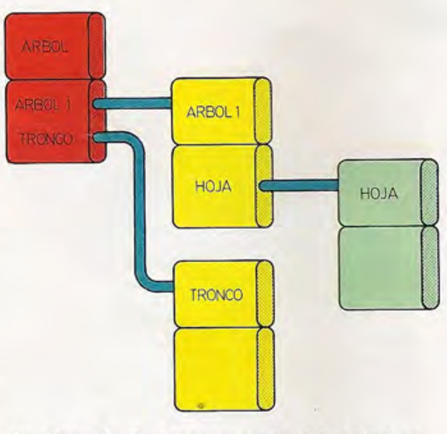
Las bases conceptuales del LOGO son la sencillez, unida a la potencia de operación. Otras características fundamentales del LOGO se concretan en su naturaleza de lenguaje modular e interactivo. La modularidad se refleja en el hecho de que el usuario del LOGO va creando pequeños módulos independientes que, a su vez, servirán de piezas para construir estructuras más complejas. El LOGO es un lenguaje interactivo en el sentido de que cualquier orden es procesada de inmediato, con sólo introducirla en el ordenador.

Debido a sus orígenes, inspirados en un lenguaje para el tratamiento de listas de datos —el LISP (LISt Processor)—, uno de los puntos fuertes del LOGO reside en la manipulación de listas alfanuméricas. No obstante, su característica más espectacular hay que buscarla en el tratamiento de gráficos, tarea encomendada al sistema denominado TURTLE GRAPHICS.

El popular TURTLE GRAPHICS (gráficos creados por medio de la tortuga), es un método fácil y divertido para la confección de dibujos. El concepto de *tortuga* procede de ciertos robots desarrollados a principios de 1960. Estas máquinas consistían en un caparazón flanqueado por ruedas (de ahí el nombre de tortuga) y funcionaban asociadas a un ordenador. El ordenador guiaba a la tortuga y ésta iba dejando marcada la huella de su trayectoria en el suelo.



La popular «tortuga» es el símbolo del lenguaje LOGO. Este simpático personaje es el protagonista del «Turtle Graphics»: un método sencillo y divertido para la creación y tratamiento de dibujos.



La filosofía de programación del LOGO se fundamenta en los denominados «procedimientos». La potencia de este lenguaje brota de la posibilidad de construir programas evolutivos, integrando procedimientos ya definidos dentro de otros más complejos.

A principios de 1970 surgió la variante actual de tan simpático y útil personaje: la tortuga que evoluciona sobre la pantalla. La idea de guiar a una tortuga a través de la pantalla de visualización, ha sido adoptada por otros muchos lenguajes para resolver el aspecto de creación de presentaciones gráficas. Junto a las posibilidades gráficas, el LOGO aporta un potente lenguaje de programación de tareas, fácil de aprender y edificado a partir de un vocabulario y unas estructuras simples y versátiles.

La sencillez inherente al empleo del LOGO, han convertido a este lenguaje en idóneo para labores pedagógicas y de iniciación a la programación de ordenadores. Los seguidores del LOGO declaran que se trata de «un lenguaje para aprender»; afirmación no sólo aplicable a *aprender a programar,* sino también extensiva a *aprender a discurrir* en base a estructuras lógicas.

En efecto, el LOGO es algo más que un lenguaje de programación; hay que concebirlo como un verdadero *entorno para la enseñanza asistida por ordenador.* Con este lenguaje, los niños aprenden a trabajar con ordenadores como si se tratara de un simple juego. El secreto reside en la naturaleza interactiva del diálogo, que facilita la creación y depuración de estructuras por el sistema de prueba-error.

Existen en el mercado distintas versiones de LOGO. La mayor parte, desarrolladas por fabricantes de ordenadores en colaboración con el Grupo Logo del M.I.T. (TI Logo, Apple Logo, Atari Logo, etc.). Se diferencian en muy poco, concretamente en abreviaturas de palabras LOGO, en la inclusión del TURTLE GRAPHICS y en el tratamiento de listas. También cabe mencionar la existencia de una versión en castellano del TI Logo (versión de Texas Instruments).

Glosario

INTELIGENCIA ARTIFICIAL: rama de la ciencia que investiga la simulación de una auténtica inteligencia en un ordenador.

DEPURACION: fase del desarrollo de un programa consistente en la búsqueda de los fallos cometidos al codificarlo o escribirlo.

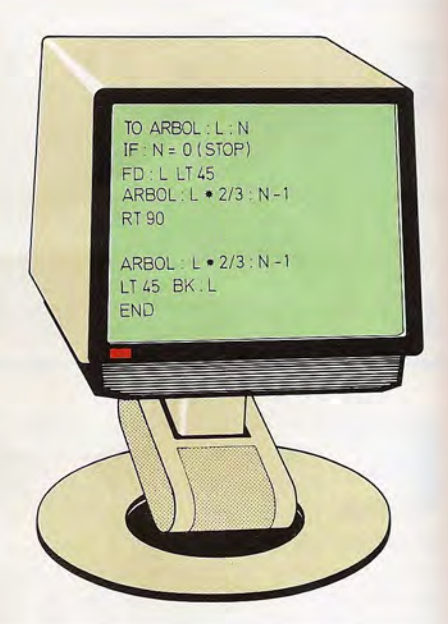
PROCEDIMIENTOS: subprogramas que forman parte de un programa mayor o «principal».

MEMORIA CENTRAL: zona para el almacenamiento de información residente en el interior del ordenador y a disposición directa de la unidad central de proceso. En ella se almacenan datos, variables y el programa o programas en curso.

LA FILOSOFIA DEL LOGO

La programación en LOGO se fundamenta en el empleo de *procedimientos* definidos por el usuario. Estos procedimientos son conjuntos de órdenes a los que se accede como si se tratara, realmente, de una orden única. A su vez, las





Aspecto de un procedimiento LOGO, o conjunto de instrucciones al que se accede a través de una orden única.

Traductores de lenguajes

La comunicación entre dos personas que hablan distinto idioma no puede establecerse sin la colaboración de un traductor. Este puede ser un intérprete que realice una traducción simultánea, frase a frase; o simplemente un traductor que escriba en el idioma del destinatario el texto redactado en el idioma de origen. Esta es una realidad trasladable al mundo de los ordenadores. Excepto en el caso de que el diálogo se mantenga utilizando el lenguaje máquina, es necesario un proceso de traducción para que el programa confeccionado por el usuario sea inteligible para el ordenador. Por supuesto, la tarea de traducción no correrá a cargo de traductores humanos; de ello se ocuparán los ordenadores una vez «instruidos» al efecto.

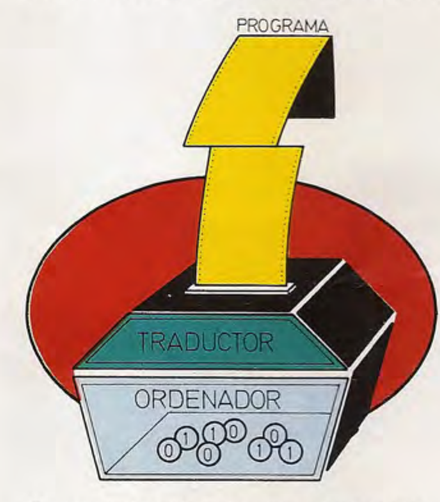
En cualquier proceso de traducción intervienen dos programas:

- *El programa fuente,* redactado en un lenguaje evolucionado, ensamblador o de alto nivel.
- *El programa objeto* o programa resultante del proceso de traducción, escrito en el lenguaje propio de la máquina.

Tal como ocurría en la traducción de comunicaciones humanas, también en este caso cabe una doble posibilidad: recurrir a un intérprete o traductor simultáneo, que permitirá un diálogo o comunicación interactiva, o utilizar los servicios de un traductor que reescriba el contenido de la comunicación en el lenguaje del destinatario (la máquina en nuestro caso).

Los eficientes traductores humanos serán ahora programas auxiliares especializados en tal menester. Y, de nuevo, especializados en realizar una traducción interactiva o demorada (del mensaje en bloque).

Los *intérpretes* son los programas especializados en la traducción interactiva.



La comunicación entre dos personas que hablan distinto idioma no es posible sin la intervención de un traductor. Algo semejante ocurre a la hora de dialogar con las máquinas. Es preciso contar con un traductor que convierta al lenguaje íntimo de la máquina las órdenes y mensajes formulados por el usuario en un lenguaje de alto nivel.

Traducen el programa línea a línea, de tal forma que el ordenador las ejecuta a medida que va disponiendo del resultado de la traducción.

La traducción diferida corre a cargo de los denominados programas *compiladores*. Estos traducen el programa fuente en bloque, obteniendo el correspondiente programa objeto redactado en el lenguaje de la máquina que debe ejecutarlo.

De la realidad humana podemos extraer otras conclusiones aplicables a los programas intérpretes y compiladores. En principio, la utilidad del intérprete se manifiesta en los diálogos interactivos; de ahí que este tipo de traductor resulte idóneo cuando se trata de habilitar una comunicación inmediata con la máquina. Otro factor significativo es que, dado el método de traducción, el intérprete invertirá bastante más tiempo en realizar su función que un compilador. Este último realiza la traducción del programa en bloque, de una sola vez, sin aguardar a que vayan ejecutándose las instrucciones a medida que son traducidas. La característica de velocidad se inclina, pues, hacia los compiladores. Este es un dato extensivo al proceso de ejecución: la ejecución del programa objeto, traducido en su integridad (compilado), es mucho más rápida que la ejecución línea a línea del programa interpretado (de tres a veinte veces más rápida).

La interpretación de un programa fuente se efectúa en el propio ordenador que cursará su ejecución. Sin embargo, el compilado de un programa puede no

órdenes están constituidas por cadenas de datos, operadores y comandos. La potencia del LOGO brota de la posibilidad de utilizar procedimientos dentro de otros procedimientos. Semejante característica, permite programar potentes operaciones partiendo de unidades elementales. El método a seguir consiste, sencillamente, en construir los *procedimientos* complejos a partir de la asociación de otros procedimientos más simples definidos con anterioridad.
En realidad, el LOGO carece del concepto



Tras el BASIC, el LOGO es el lenguaje más popularizado en el terreno de los equipos domésticos. Son muchos los ordenadores personales de esta categoría que disponen de un intérprete de lenguaje LOGO.

realizarse en el ordenador que debe ejecutar el programa objeto. Es frecuente que sea un ordenador auxiliar el que se encargue de generar el programa objeto, redactándolo en el lenguaje máquina propio del equipo al que vaya destinado. Terminada la compilación, el programa puede ya introducirse y ejecutarse cuantas veces sea necesario en el ordenador de destino.
Son muchos los lenguajes para los que se dispone de ambos tipos de traductores.

La elección de intérprete o compilador dependerá del tipo de actividad que se encomiende al ordenador. Si la velocidad no es un factor primordial y prima la necesidad de mantener un diálogo interactivo, habrá que optar por un intérprete. Los microordenadores domésticos incorporan de origen un intérprete de lenguaje BASIC; si bien, el fabricante suele disponer de un catálogo de intérpretes y compiladores, en opción, de otros lenguajes.

Cuando la característica solicitada es una alta velocidad de ejecución, no cabe duda en la elección: hay que optar por un compilador.
Los lenguajes con una marcada inclinación hacia las aplicaciones interactivas (BASIC, LOGO, PILOT...) suelen utilizarse en versión compilada; mientras que los lenguajes más especializados, habitualmente no interactivos (FORTRAN, COBOL, ALGOL, PL/3...) actúan a través de compiladores.

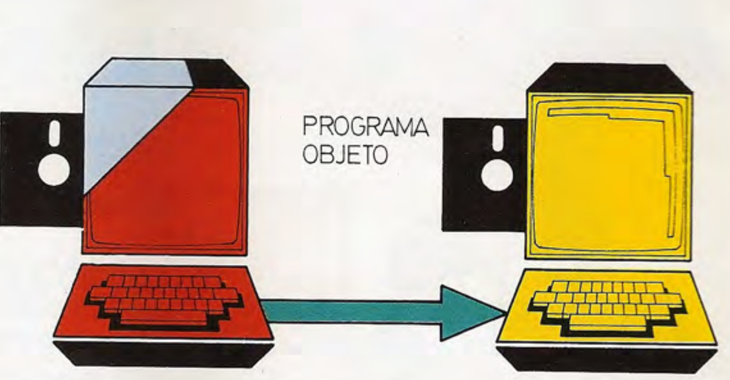
INTERPRETE DE LENGUAJE
PROGRAMA EN EJECUCION
COMPILADOR DE LENGUAJE
PROCESO DE COMPILACION

INTERPRETE



COMPILADOR

PROGRAMA FUENTE



La actividad del traductor puede manifestarse realizando una traducción simultánea del diálogo, o efectuando la traducción del mensaje total, de una sola vez. Los programas traductores del primer tipo son los denominados «intérpretes» y los del segundo «compiladores».



El nacimiento del LOGO tuvo su fuente de inspiración en el LISP, un lenguaje evolucionado orientado a aplicaciones de inteligencia artificial. Dado su origen y sus propias características, el LOGO es un lenguaje que resulta adecuado en el marco de la robótica.

de programa como tal, lo que se suple con el denominado «WORK SPACE» o *espacio de trabajo*. Este contiene los procedimientos y variantes definidos por el usuario; y, por supuesto, en él existirá, generalmente, un *superprocedimiento,* o procedimiento principal, que hace uso de otros subprocedimientos.
En un mismo *espacio de trabajo* pueden coexistir varios superprocedimientos independientes. Esto equivale a tener varios «programas» independientes en la memoria central.
Como ya se ha indicado, el LOGO es un lenguaje fundamentalmente interactivo. Los mensajes de error son claros y concisos. Por ejemplo, si a través del teclado se introduce la palabra "HOLA", seguida de una acción sobre la tecla RETURN (la orden para que el ordenador asimile el mensaje introducido), la máquina responderá con un significativo:

I DON'T KNOW HOW TO HOLA

(¡Ignoro cómo realizar la acción HOLA!)

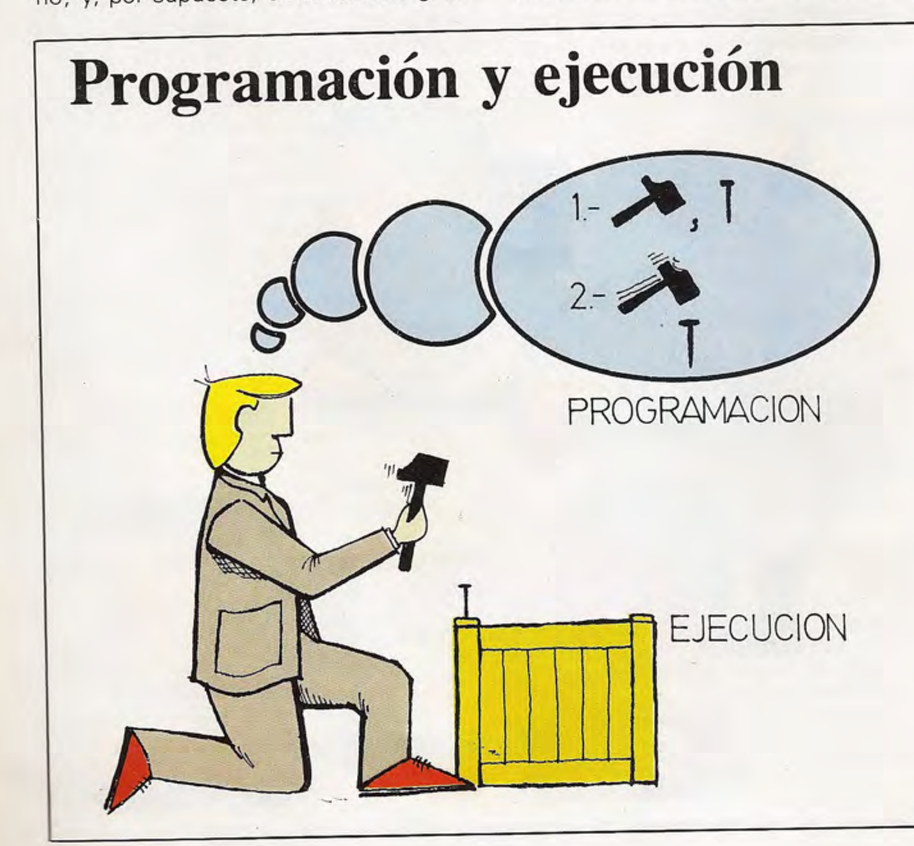
Con este mensaje, el LOGO comunica al usuario que el procedimiento HOLA no está definido en el *espacio de trabajo*. Si el error deriva de la omisión de un dato de entrada dentro de una orden, el LOGO responderá esta vez con:

NOT ENOUGHT INPUTS TO XXX

(¡No hay suficientes datos de entrada para cursar la orden XXX!)

Otra capacidad notable del LOGO se manifiesta en el tratamiento de cadenas alfanuméricas. Sus potentes instrucciones permiten añadir, extraer y comparar elementos inmersos en las mismas sin ninguna dificultad.

Programación y ejecución



Los lenguajes informáticos comparten el mismo objetivo que los humanos: ofrecer el medio adecuado para que pueda establecerse una comunicación. En el caso de la máquina, la comunicación se concreta en «instruirla» para que ésta desarrolle un trabajo o realice una tarea específica. Aquí es donde aparece el concepto de programa «o secuencia ordenada de instrucciones», cuya puesta en práctica o *ejecución*
resuelve un cálculo, efectúa un tratamiento de información o, en general, realiza la tarea detallada en el programa.
En definitiva, *programar* al ordenador equivale a confeccionar la secuencia de instrucciones o programa, utilizando un lenguaje apropiado, inteligible para el ordenador.
Cuando la máquina deba efectuar el conjunto de operaciones o tarea programada, es preciso que el usuario introduzca el programa en la memoria del ordenador. A partir de ese instante, éste puede ordenar su *ejecución*. Ejecución que realizará el ordenador examinando las sucesivas instrucciones que componen el programa, interpretando su significado y cursando las órdenes y operaciones encomendadas.

Los S.Os. de la microinformática

Del nacimiento del microprocesador a los modernos sistemas operativos

EL desarrollo de los sistemas operativos ha seguido un camino paralelo al de la evolución de los ordenadores. Es evidente, pues, que al nacer el microprocesador y, en torno a éste, desarrollarse todo el universo microinformático, iba a manifestarse una evolución semejante en el terreno de los sistemas operativos.

Los albores de la microinformática, representados por los primeros modelos de las firmas americanas Apple Computers y Radio Shack, o por el alabado PET de la también americana Commodore, no tuvieron un impacto inmediato en el terreno de los S.Os.

Los primeros microordenadores no disponían de un sistema operativo organizado y con entidad propia. Todos ellos incluían un escueto y primitivo repertorio de funciones básicas para el control de la máquina, integrado dentro del traductor de lenguaje BASIC.

Este método aún está presente en ciertos ordenadores domésticos (ZX-81, ZX-SPECTRUM, ORIC...). El intérprete del lenguaje BASIC incorpora algunas funciones elementales propias de cualquier sistema operativo; por ejemplo, destinadas al control de los periféricos asociados al equipo: pantalla de visualización, grabador/reproductor de casetes...

Tal como ocurre en los ordenadores personales menos evolucionados, los primeros microordenadores encerraban una total exclusividad en cuanto al uso de programas. Toda la estructura de programación dependía del propio intérprete, que no sólo ejercía la tarea de traductor, sino que también gobernaba los recursos del hardware.

Este método de explotación hace imposible la confección de programas utilizables en distintas máquinas; puesto que la compatibilidad de los programas sólo será real cuando las máquinas compartan, además del intérprete, toda la estructura circuital controlada por el mismo.

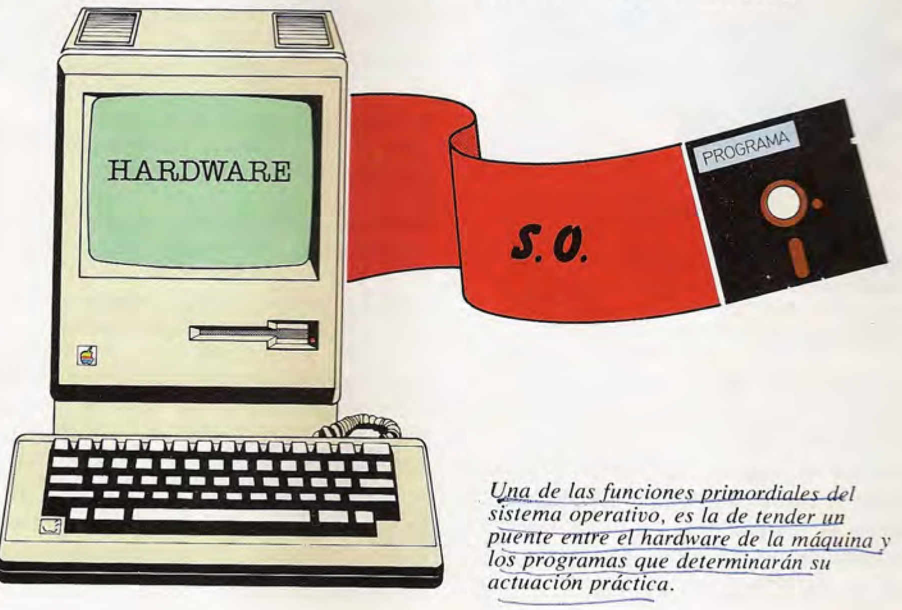
NACIMIENTO DEL CP/M

Esta deficiencia incitó a algunos expertos a lanzarse en busca de un sistema operativo para los nuevos equipos, pioneros de la microinformática. Era preciso desarrollar un sistema operativo capaz de actuar como puente entre el hardware de distintas máquinas y el traductor de lenguaje; de lograrlo podía avanzarse en la tan anhelada senda de la estandarización.

De ahí nació el CP/M *(Control Program for Microprocessors:* programa de control para microprocesadores), de manos de la compañía americana Digital Research. Tras éste fueron brotando el Apple.Dos y otros tantos sistemas operativos, casi todos ellos con una denominación terminada con las siglas DOS *(Disc Operating System:* sistema operativo en disco).

Durante algunos años, el CP/M fue el sistema operativo casi exclusivo en el campo de los microprocesadores. En 1973, la firma Intel anunció el desarrollo del primer microprocesador con potencia suficiente como para constituir la unidad central de proceso de un microordenador. A partir de ese instante, Gary Kildall empezó a concebir las primeras versiones de lo que, más adelante, se convertiría en el popular CP/M. Un sistema operativo en plena vigencia y que, en la actualidad, puede encontrarse en cerca de un millón de ordenadores.

Al nacer el CP/M, los fabricantes se encontraron ante la alternativa de acogerse al mismo, o crear un sistema operativo exclusivo para su microordenador. Las contrapartidas eran obvias: el uso del CP/M abría las puertas para que su equipo pudiera utilizar cualquier programa adaptado a este sistema operativo.



Una de las funciones primordiales del sistema operativo, es la de tender un puente entre el hardware de la máquina y los programas que determinarán su actuación práctica.

La cada vez mayor aceptación del CP/M, revirtió en el nacimiento de una biblioteca de programas que podían ejecutarse en distintos microordenadores. Poco a poco, el repertorio de programas fue incrementándose con las aportaciones de los fabricantes y empresas de programación que adoptaron el nuevo estándar. Así es como el CP/M se convirtió en el líder indiscutible de los sistemas operativos, en la escasa década de vida de la microinformática.

DEL MICROPROCESADOR AL S.O.

Si el microprocesador es el cerebro integrado que ha permitido el nacimiento de los microordenadores, parece obvio que también será éste el responsable de la trayectoria microinformática de los sistemas operativos.

MICROPROCESADOR
Z-80
6502
8086
8088
68000
CP/M
OASIS
MP/M
MSX-DOS
Pro DOS
DOS 3.3.
SOS
CP/M-86
MS/DOS
UNIX
PICK
SISTEMA OPERATIVO

Su relación con la intimidad de la máquina llega hasta tal punto que los sistemas operativos están diseñados para un determinado tipo de microprocesador.

Una de las funciones primordiales del sistema operativo es la de actuar de puente entre el hardware de la máquina y los programas de aplicación. En consecuencia, el sistema operativo tiene que estar concebido en perfecta consonancia con el microprocesador. Cada sistema operativo está destinado a un determinado tipo de microprocesador, o a una familia de microprocesadores que comparten unas características comunes. Así, por ejemplo, el sistema operativo ProDOS está destinado al microprocesador 6502, mientras que el MS/DOS está creado para coexistir con el microprocesador 8088.

En esta primera década de la microinformática, la época de los microprocesadores de 8 bits, el liderazgo ha correspondido al sistema operativo CP/M. Una de las bazas que más ha contribuido a la proyección del CP/M, hay que buscarla en la familia de microprocesadores para los que está destinado: 8008, 8080, 8085 y Z-80.

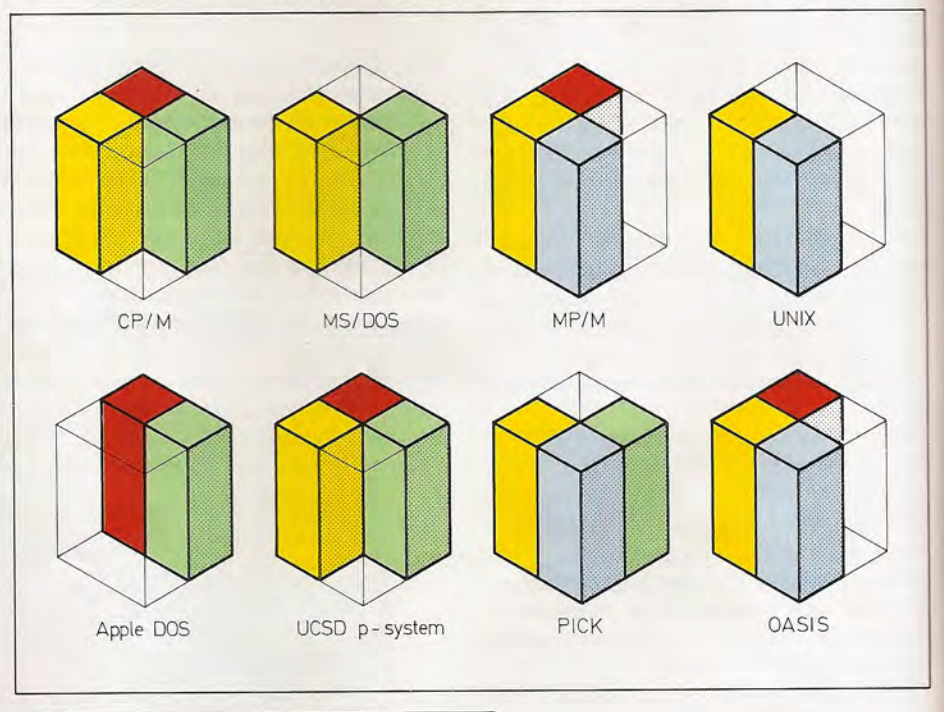
Hay que tener en cuenta que los principios del CP/M están ligados al propio nacimiento del microprocesador. La firma americana Intel —uno de cuyos técnicos fue, precisamente el creador del CP/M, Gary Kildall— fue quien puso en el mercado el primer microprocesador. Un avance que se consumó con el lanzamiento del primer microprocesador de 8 bits, el 8008, y con el desarrollo del primer microprocesador capaz de constituir el cerebro de un microordenador: el Intel 8080. Esta línea tuvo su posterior continuidad en el 8085 y llegó al pleno dominio del mercado con el Zilog Z-80: un microprocesador que, aun sin salir de las manos de Intel, perpetuaba la filosofía de los productos de esta firma en el terreno de los microprocesadores de 8 bits.

El predominio de esta familia de microprocesadores de 8 bits, constituye una razón de verdadero peso para justificar el alcance que ha logrado el sistema operativo CP/M.

S.Os. PARA MICROPROCESADORES DE 8 BITS

El absoluto protagonismo del CP/M, ensombrece a otros sistemas operativos cuya potencia es, en algunos casos, plenamente equiparable a la del líder. Entre éstos se encuentran el OASIS, Apple. DOS, PICK, FLEX, TURBODOS, ProDOS y UCSD p-System.

Con el empuje de los microordenadores de 8 bits, han ido apareciendo nuevas versiones, cada vez más actualizadas, del





Características básicas de los sistemas operativos más importantes en el terreno de la microinformática.

El microprocesador es el cerebro integrado que ha permitido el nacimiento de la microinformática. Un protagonismo que lo convierte en el responsable de la evolución de los modernos sistemas operativos.



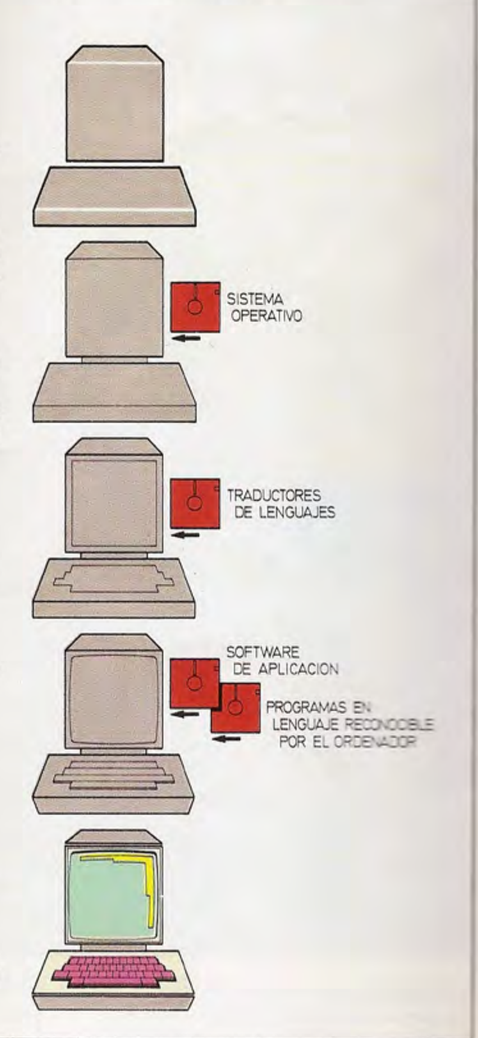
CP/M-80. La más utilizada en la actualidad es la revisión 2.2 (CP/M 2.2).
Recientemente, ha aparecido un nuevo sistema operativo que promete alcanzar una implantación sustantiva en el terreno de los microprocesadores de 8 bits. Este es el denominado MSX.DOS, de la compañía americana Microsoft. Además de estar destinado a un microprocesador compartido por el CP/M, el Z-80, el MSX.DOS presenta una cierta compatibilidad con el protagonista; por ejemplo, puede utilizar archivos creados a partir del CP/M.
El objetivo del MSX.DOS se sitúa en los ordenadores personales de tipo familiar. Constituye uno de los elementos que configuran la norma MSX, a la que se han adscrito más de una docena de fabricantes japoneses y algún europeo.
Tras los microprocesadores de 8 bits, llegaron los microprocesadores capaces de operar internamente con palabras binarias de 16 bits. Actualmente, existen ya microprocesadores de 32 bits. La evolución de los ordenadores personales ha seguido por completo la línea trazada por los microprocesadores. Hasta tal punto que la mayor parte de los equipos profesionales y de gestión incorporan un microprocesador de este tipo.
Los 8 bits han quedado relegados casi por completo al terreno de los ordenadores domésticos, con un acentuado dominio del microprocesador Z-80, seguido a distancia por el 6502 y, tras éste, el 6809.

EL SALTO A LOS 16 BITS

El liderazgo del CP/M se diluye en el marco de los microordenadores de 16 bits. La irrupción de la multinacional IBM en el mundo del ordenador personal, supuso el vertiginoso ascenso de un nuevo sistema operativo que supera en este ámbito al CP/M. Se trata del MS/DOS, creado por la firma americana Microsoft. El MS/DOS, rebautizado en los ordenado-

De máquina a ordenador

El primer contacto con la idea de ordenador suele derivar de la observación de una máquina que, con rapidez y eficacia, confecciona la nómina de la empresa en la que trabajamos o que, sencillamente, llena los ratos de ocio con emocionantes juegos en la pantalla del televisor.
Para llegar a este nivel de funcionalidad, hay que equipar al hardware o arquitectura física de la máquina con un surtido de vituallas que lo irán convirtiendo en la eficaz herramienta que gestiona una aplicación. El trayecto de máquina inerte a ordenador pasa por tres etapas básicas:
a) Incorporación del sistema operativo.
b) Con la máquina dotada de una inteligencia básica, puede ya pensarse en completarla con un traductor de lenguaje que facilite un diálogo más completo y directo.
c) La máquina está ahora en disposición de interpretar y ejecutar programas de aplicación creados por el usuario o programas adquiridos para instruir a la máquina en cierta tarea específica. El *software de aplicación,* o paquetes de programas que permiten al ordenador realizar una tarea específica, suele nacer orientado a su complementariedad con el sistema operativo. Por ello, en muchos casos, puede ocupar directamente la segunda planta del edificio informático, instalándose sobre el nivel ocupado por el sistema operativo.



res personales IBM como PC/DOS, se ha convertido, en muy poco tiempo, en un sistema operativo de gran popularidad y cuya implantación crece día a día.
Casi todos los sistemas operativos que coexistían en el mercado de los equipos de 8 bits, han derivado de nuevas versiones adaptadas a los microordenadores de la nueva generación. Tal es el caso del propio CP/M, que ofrece las versiones CP/M-86 y Concurrent CP/M, para los microprocesadores 8086 y 8088, y el CP/M-68K destinado al microprocesador 68000 de la firma Motorola.
Otros sistemas operativos relevantes en el campo de los 16 bits son el UNIX, OASIS, PICK y UCSD P-system. En otro rango están los sistemas operativos que crea el propio fabricante para sus equipos. A este grupo pertenece, por ejemplo, el QL-DOS destinado al Sinclair QL.

S.Os. MULTIUSUARIO E INTEGRADOS

Otra consecuencia de la evolución de los microordenadores, es la posibilidad de trabajo en régimen «multiusuario». El ordenador reparte su atención entre varios usuarios, cursando en cada caso un proceso distinto; no obstante la velocidad es tal que los tiempos de espera son casi insignificantes.
Los sistemas operativos para microordenadores han entrado en los equipos multiusuario, aportando funciones propias de sistemas operativos para miniordenadores y grandes equipos. De nuevo, predominan en este ámbito algunas versiones de sistemas operativos monousuario. Por ejemplo, los MP/M-80 y MP/M-86, que son versiones multiusuario del CP/M destinadas a equipos de 8 y 16 bits, respectivamente. También cabe destacar al UNIX, un sistema operativo multiusuario, para 16 bits, de reconocida potencia. La firma Microsoft, que detenta la autoría del MS/DOS, también desarrolló su sistema operativo multiusuario, para 16 bits, denominado XENIX. E incluso el sistema operativo OASIS se encuentra en versiones multiusuario para microordenadores de 8 y de 16 bis.
El último peldaño en los sistemas operativos para microordenadores, lo ocupan los sistemas operativos integrados. Con ellos se diluye la separación entre sistema operativo y software de aplicación. Suponen un cambio en la filosofía de trabajo de los ordenadores, ofreciendo al usuario un entorno multi-tarea caracterizado por una gran facilidad de control. En este grupo se inscriben los sistemas operativos de los modelos Lisa y Macintosh de la firma Apple Computers, o el sistema operativo integrado del ICL Perq.
La opinión de los expertos vaticina un prometedor futuro para esta nueva generación de sistemas operativos, hasta el punto de que en pocos años, pueden llegar a sustituir por completo a los S.Os tradicionales.



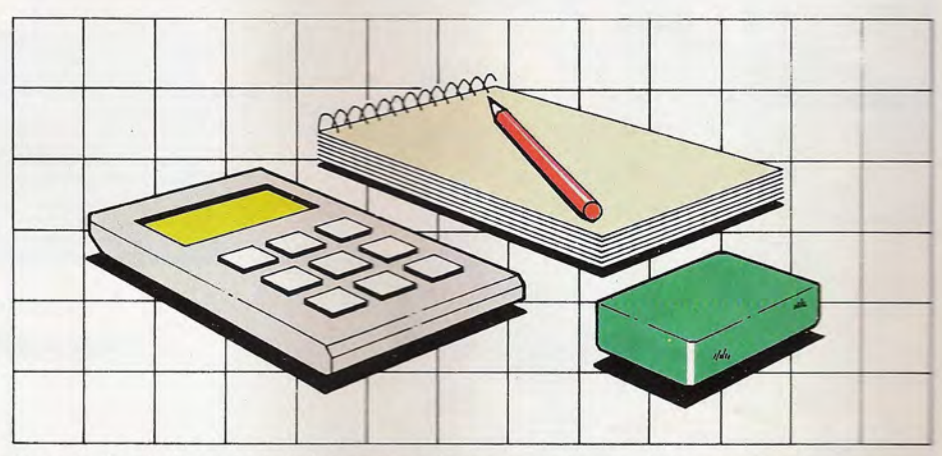
Los sistemas operativos integrados constituyen la más reciente innovación en el terreno de los sistemas operativos. Su presencia en el ordenador aporta una nueva filosofía de trabajo que facilita la relación hombre/máquina.

En busca del ordenador...

A través del software de aplicación

DE nuevo, hay que partir de la constatación de que el rendimiento del ordenador en el terreno práctico depende, casi por completo, de la calidad de los programas que lo «instruyan»: del software de aplicación. Ante tal realidad, no cabe duda de la importancia que tiene elegir el software idóneo para cada aplicación.

¿COMO ELEGIR EL SOFTWARE DE APLICACION?



La entrada en el mundo de la informática, deriva, normalmente, de la necesidad de contar con un auxiliar eficiente y capaz de resolver determinadas tareas. Una vez decidida la aplicación o aplicaciones que hay que automatizar...

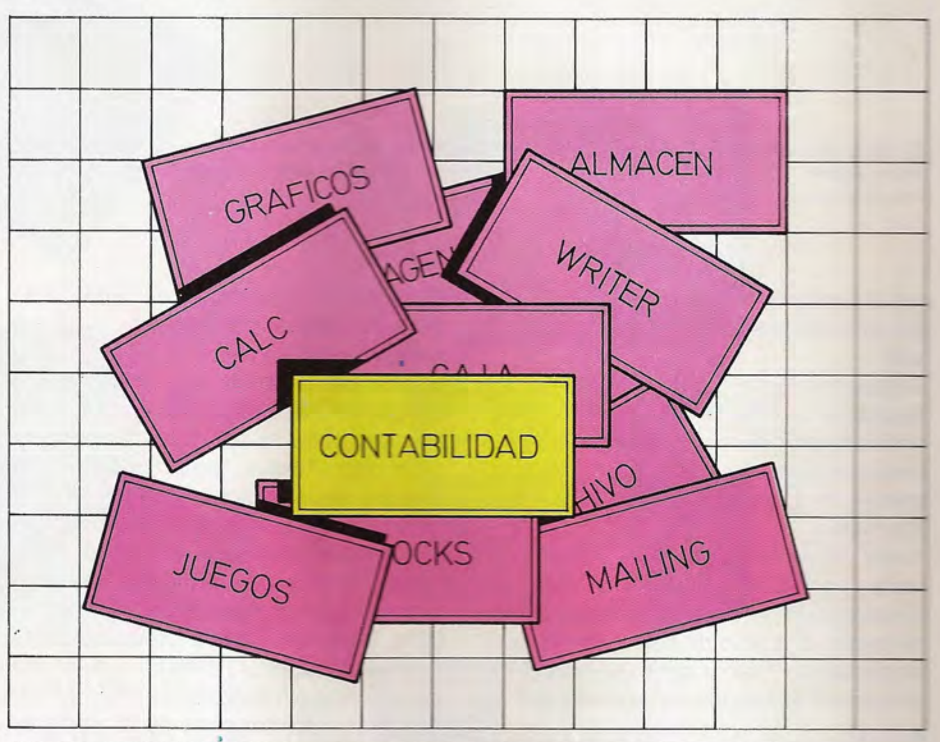
Varios son los factores que entran en juego a la hora de elegir el software. Factores que van desde la evaluación estricta de la tarea a resolver, hasta la elección del soporte adecuado en cada caso. Aun cuando se prestará una especial atención al tema más adelante, es conveniente anticipar en este punto algunas consideraciones que deben intervenir en la elección del software idóneo.

1 *Una definición precisa y detallada de las necesidades*

Este es el primer paso, ineludible para la correcta elección. Cuanto más precisa sea la definición de las características de la tarea o aplicación que se desea informatizar, mayor será la garantía de éxito en la decisión final. Una exposición detallada y exacta permitirá descartar programas que, aun ajustándose algunas de las características solicitadas, omiten la resolución de tareas que limitan el rendimiento práctico de la aplicación.

2 *¿Software de creación propia, «a medida» o estandarizado?*

El nivel de conocimientos de programación por parte del usuario, la complejidad de la aplicación y la existencia de paque-



Hay que comprobar si existe en el mercado de algún programa ajustado a nuestras necesidades específicas.



El siguiente eslabón que conducirá al ordenador idóneo, lo constituye la respuesta a un nuevo interrogante: ¿Para qué sistema operativo está concebido el programa o paquete de aplicación seleccionado?

tes de aplicación estandarizados, son algunos datos que intervienen en esta decisión.

El desarrollo del mercado del software, ha llegado a un nivel en el que es posible encontrar programas de cualquier tipo y a precios moderados; de ahí que, habitualmente, no suela contemplarse la primera alternativa.

Antes de decidirse por encargar un software «a medida», es necesario examinar con detalle la oferta de aplicaciones estandarizadas. El precio de estas últimas será siempre muy inferior al de una aplicación semejante encargada con exclusividad.

3. *¿Qué soporte es el más adecuado?*

Esta es una decisión que dependerá de las propias características del ordenador y de la naturaleza del programa o del paquete de aplicación.

El propietario de un ordenador personal de tipo familiar, suele contar con tres posibles soportes: casete, cartucho enchufable o disco flexible. Los programas destinados a estos equipos —de juegos, educativos o de gestión personal— suelen ofrecerse almacenados en más de un soporte. Siempre que exista tal posibilidad, la opción preferible es el cartucho enchufable. En primer lugar, la velocidad de trabajo será superior, puesto que el programa viene almacenado en una memoria que pasa a formar parte de la propia estructura direccionable por el ordenador. La comodidad también será superior: no hay que realizar operaciones de carga o trasvase del programa del soporte a la memoria interna del equipo. Tan sólo hay un dato en contra: el precio del cartucho es superior al de la casete o del disco.

Las aplicaciones más complejas, científicas, de gestión o administrativas, eluden, normalmente, cualquier otro soporte distinto del disco.

4. *Compatibilidad con el ordenador*

Los programas y paquetes de aplicación no son universales, sino que están concebidos para un determinado equipo, o para su compatibilidad con un sistema operativo específico.

Una misma aplicación puede ofertarse en distintas versiones. Por ejemplo, la popular hoja electrónica «Visicalc» está disponible para distintos equipos familiares y para varios sistemas operativos habituales en el campo de los ordenadores personales más evolucionados (CP/M, MS/DOS, OASIS...).

A la hora de elegir el paquete adecuado, habrá que verificar su total compatibilidad con el ordenador destinatario, tanto por lo que respecta al sistema operativo como al soporte en el que se entrega. El disco flexible que almacena el paquete de aplicación, debe ser compatible con la unidad de disco asociada a nuestro ordenador.

5. *La calidad de la documentación*

Un factor muy importante es la amplitud, detalle y tratamiento de la documentación que acompaña a la aplicación. No hay que perder de vista que el usuario debe familiarizarse con la aplicación a través del manual.

Excepto en el caso de los programas de juego, que no precisan de excesivos comentarios, el manual debe ser completo, detallado y redactado en un idioma que domine el usuario. Este llegará a extraer un rendimiento idóneo en la medida en que aprenda a utilizar la aplicación y conozca todas y cada una de las posibilidades que le brinda.

También es conveniente que la documentación incluya una relación de posibles fallos, incluso debidos a omisiones del propio usuario no experimentado, con un claro detalle de las soluciones en cada caso.

Cada día son más las aplicaciones que sustituyen la amplitud del manual escrito por un programa de aprendizaje, de tipo tutorial. Esta es una opción cómoda y apreciable.

El ordenador personal: una gran familia

La actividad del ordenador personal se manifiesta en aplicaciones que van desde la simple generación de un juego en la pantalla del televisor, hasta la gestión completa de las tareas administrativas de una pequeña o mediana empresa. Adoptando criterios exclusivamente prácticos, como puede ser el volumen y la potencia de trabajo, cabe establecer una clasificación dentro del mundo de los ordenadores personales. Una clasificación plagada de intersecciones entre los diversos grupos, y que permanece abierta, a tenor de la constante evolución de estas máquinas y a la proliferación de modelos de casi cualquier volumen y posibilidades:

- Ordenadores de bolsillo
- Ordenadores domésticos
- Ordenadores portátiles
- Ordenadores profesionales
- Ordenadores de gestión

El recinto en el que se plasma la actividad de estos grupos de ordenadores personales es, ni más ni menos, el que inspira su denominación. Desde el simple ordenador de bolsillo (como es el caso del ZX-81), hasta el consumado ordenador de gestión (Apple Lisa, HP-150, NCR DM-V o DEC Professional-300), se encuentra un extenso abanico de equipos, más o menos potentes y capaces de automatizar las tareas más diversas.





ORDENADORES DE BOLSILLO



ORDENADORES PROFESIONALES



ORDENADORES DOMESTICOS





ORDENADORES PORTATILES





ORDENADORES DE GESTION

6. *El precio*

No podía faltar el condicionante económico. Como se indicó en un párrafo anterior, los paquetes de aplicación estandarizados resultan mucho más económicos que sus equivalentes confeccionados por encargo. En todo caso, es obvio que tal economía sólo estará al alcance si existe una aplicación estándar que satisfaga las exigencias impuestas por el usuario.

DE LA APLICACION AL ORDENADOR

La decisión de entrar en el universo de la informática parte, casi siempre, de la necesidad de contar con un colaborador rápido y eficiente, que solucione una o múltiples tareas. Por supuesto, la entrada en la informática también puede tener su origen en el interés por contar con un versátil experto que amenice los ratos de ocio con los más diversos juegos.

En uno y otro caso, el origen está en la aplicación: en encontrar a un auxiliar que resuelva la papeleta de confeccionar doscientas cartas, cada una dirigida a un determinado cliente, en reemplazar a la calculadora y el lápiz por un cuaderno electrónico que calcule y actualice los supuestos que reflejamos en sus casillas, o en reclutar a un jugador, experto y versátil, que llene de acción los ratos de ocio.

Estas son situaciones que revelan el predominio real de la tarea a resolver sobre la máquina que se ocupará de ponerla en práctica. El usuario llegará a conocer a su ordenador a través de los programas de aplicación. Y de éstos dependerá habitualmente las prestaciones de la máquina. Esta es una realidad que evidencia, a todas luces, la importancia de los programas de aplicación.

No ha de resultar ajeno, pues, que un administrativo piense antes en las características del programa que automatice su contabilidad, que en el tipo de microprocesador, o en el número de conectores para comunicación externa, que incorpora el ordenador que debe adquirir. Estas últimas serán casi meros accidentes desde su perspectiva.

También es lógico que un estudiante, cuyo objetivo es repasar cómodamente los conceptos matemáticos y cazar naves espaciales en los ratos de descanso, opte por adquirir un equipo doméstico, antes que un sofisticado y potente ordenador de gestión.

La vía para informatizarse parte, habitualmente, del nivel de la aplicación, y recorre, en orden inverso, los restantes estratos del edificio informático: sistema operativo y hardware del ordenador.

En definitiva, una vez que el futuro usuario ha decidido cuáles son las tareas que debe resolverle la máquina, debe ir respondiendo a las siguientes preguntas que le ayudarán a precisar la elección del ordenador personal idóneo:

- *¿Existe algún programa o paquete de programas en el mercado, capaz de resolver, en todos sus matices, la aplicación?*

- *Si existe, ¿para qué sistema operativo está concebido el paquete de aplicación que se considera idóneo para solventar sus necesidades?*

- *¿Qué ordenadores son los que incorporan o admiten el sistema operativo en cuestión?*

Y, por último, queda la labor de seleccionar el ordenador adecuado entre los que pueden ejecutar la aplicación deseada. Para precisar la elección, es conveniente evaluar en este punto otros condicionantes; por ejemplo:

- *¿Los ordenadores considerados son compatibles físicamente con el soporte de la aplicación?*

¿El disco en el que se entregan los programas es compatible con la unidad de disco del ordenador? ¿La máquina dispone de suficiente memoria residente para ejecutar la aplicación? ¿Es posible conectar al ordenador los periféricos idóneos al caso (impresora, terminal, digitador, trazador gráfico...)?...

Todo un repertorio de cuestiones cuya respuesta conducirá a la elección del ordenador idóneo para resolver nuestras necesidades de automatización.



Una vez que se conoce cuál es el sistema operativo con el que coexiste el software de aplicación elegido, queda ya precisada la gama de equipos entre los que estará la decisión final.

Los datos del BASIC

Ejecución de programas y recolección de datos

Un primer contacto con la realidad del lenguaje BASIC lo proporciona el uso del comando PRINT. En sus distintas formulaciones, este comando permite trasladar información al órgano que refleja, a ojos del usuario, la actividad del ordenador.
Las posibilidades del comando PRINT no se limitan a la creación de los tipos de instrucciones cuyos formatos se analizaron en el capítulo precedente. Existen otras variantes, frecuentes en muchos intérpretes BASIC, que amplían el abanico de posibilidades del PRINT.

VARIANTES DE LA INSTRUCCION PRINT

Por el momento sólo se han utilizado instrucciones PRINT formuladas de acuerdo a su formato básico, cuya expresión general es:

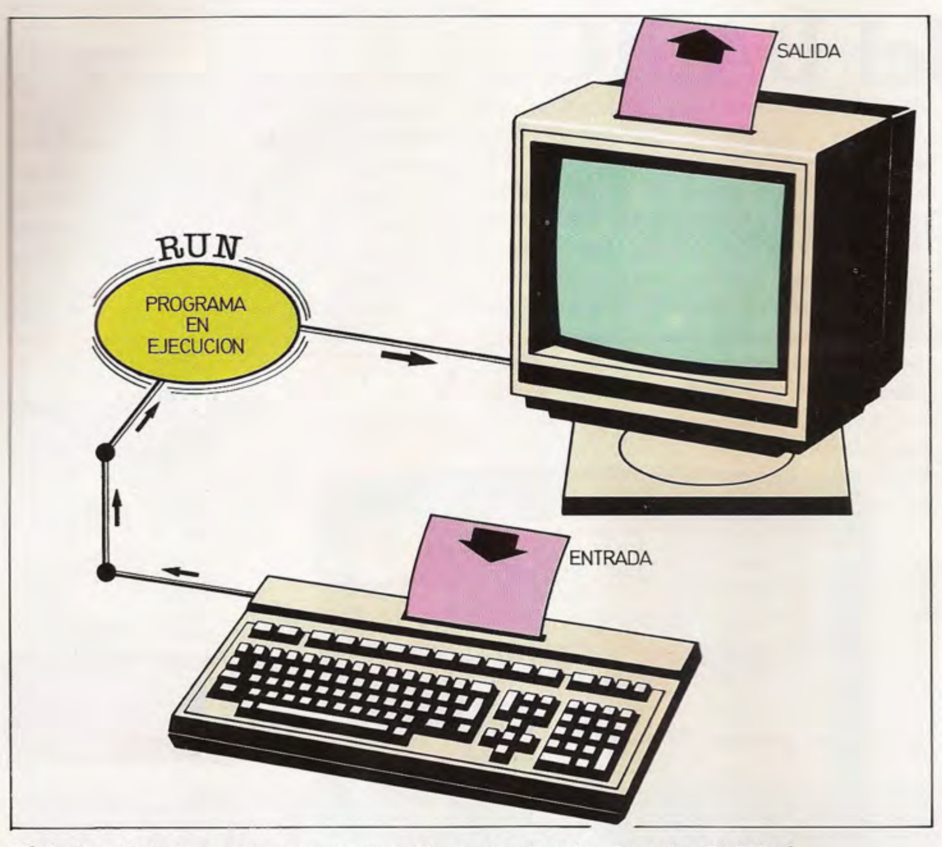
(NL) PRINT <*expresión 1*>{[;] [,]}<*expresión 2*>...

Al utilizar este formato básico como instrucción directa, hay que incluir el número de línea (NL). Este precede al comando PRINT y a su argumento que, cabe recordar, puede estar constituido por uno o varios datos o expresiones.
Ciertos intérpretes BASIC, admiten otras variantes en la formulación del comando PRINT además de la que se ha estudiado como caso general.
Una de ellas es la que obedece al siguiente formato:

(NL) PRINT AT(X,Y) <*argumento*>

Una de las variantes de la instrucción PRINT es la que incorpora la función AT(X,Y). Esta permite seleccionar el punto de impresión del texto que figura en el argumento.

VARIANTES DE LA INSTRUCCION PRINT
Formato: (Número de línea) PRINT AT(X,Y) <*argumento*> *Ejemplos:* 20 PRINT AT(20,12) "JUAN PEREZ" 40 PRINT AT(5,3) "NOMBRE:";A$ *Definición:* Escribe el argumento a partir del punto X,Y de la pantalla; siendo X el número de columna e Y el número de fila. El origen de coordenadas se encuentra en el ángulo superior izquierdo de la pantalla.
Formato: (Número de línea) PRINT TAB(N); <*argumento*> *Ejemplos:* 45 PRINT TAB(5); "TABLA" 60 PRINT TAB(25); C$,D *Definición:* Escribe el argumento a N espacios de distancia del margen izquierdo de la pantalla. Actúa de forma análoga al tabulador de una máquina de escribir.
Formato: (Número de línea) PRINT SPC(N); <*expresión 1*>: SPC(M); <*expresión 2*>... *Ejemplos:* 30 PRINT SPC(7); "ARTICULO"; SPC(15) "PRECIO" 40 PRINT SPC(10); A$; SPC(18); P *Definición:* Imprime la expresión correspondiente a N (o M) espacios a la izquierda de la posición en la que se encontraba el cursor.



El destino de todo programa es su ejecución en el ordenador. La orden BASIC al efecto es RUN.

La función AT(X,Y) que sigue al comando PRINT permite al usuario precisar el punto de la pantalla en que desea visualizar el argumento. Para ello, debe especificar los valores de X (columna: coordenada horizontal) y de Y (fila: coordenada vertical), teniendo en cuenta que el origen de coordenadas se encuentra en el ángulo superior izquierdo de la pantalla (ver figura adjunta). Por ejemplo, la instrucción siguiente:

PRINT AT(5,2) "JUAN"

escribirá la palabra JUAN en la segunda fila de texto de la pantalla y a partir de la quinta columna; o lo que es lo mismo, dejando cuatro espacios en blanco a partir del margen izquierdo de la pantalla.

Otra de las variantes de la instrucción PRINT, resulta especialmente adecuada para escribir en la pantalla dejando un determinado número de espacios en blanco; algo semejante a lo que permiten las tabulaciones de una máquina de escribir. Su formato genérico es:

(NL) PRINT TAB(N); *<argumento>*

Su utilidad es manifiesta a la hora de confeccionar tablas, puesto que dando valores fijos a N, pueden seleccionarse perfectamente las columnas en las que se realizará la presentación de los datos. Cabe indicar al respecto que el valor de N debe coincidir con el número de espacios en blanco que quieran dejarse desde el margen izquierdo de la pantalla hasta el punto de escritura del primer caracter del argumento.

Existe una tercera variante también destinada a precisar el punto de escritura sobre la pantalla. Su formato es:

(NL) PRINT SPC(N); *<expresión 1>*; SPC(M); *<expresión 2>...*

El dato o valor de la expresión que sigue a cada función SPC se escribirá tantos espacios a la izquierda de la última posición escrita como dicte el parámetro que acompaña a SPC (N, M...).

Por último, hay que constatar que determinados dialectos BASIC, permiten la sustitución de la palabra comando PRINT por el símbolo de cierre de interrogación (?). No existe diferencia alguna en el comportamiento y, realmente, la única justificación se encuentra en el intento de hacer más cómoda y rápida la escritura de las instrucciones PRINT. Por ejemplo, las dos siguientes líneas de programa son coincidentes desde el punto de vista de su ejecución:

20 PRINT "ALTERNATIVA AL COMANDO PRINT"
20 ? "ALTERNATIVA AL COMANDO PRINT"

A lo largo de la obra habrá ocasión de comprobar la utilidad práctica de todas las posibles versiones de la instrucción PRINT dentro de los programas BASIC.

EJECUTANDO EL PROGRAMA

El destino de cualquier programa no es otro que su ejecución en el ordenador. Para cursar esta orden a la máquina, existe un comando BASIC al efecto: RUN.

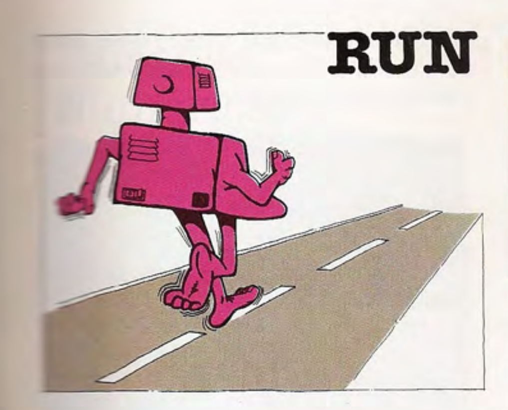
Dada su naturaleza de comando de control, RUN se utiliza a modo de instrucción directa, sin número de línea. Puede introducirse en el ordenador en cualquier instante en que el intérprete BASIC esté dispuesto para recibir un mensaje.

RUN

Orden para la ejecución del programa en curso, almacenado en memoria.

Formato: RUN *<número de línea><, R>*

Ejemplos:
RUN
RUN 26
RUN 100,R.



Su ejecución provoca un borrado inicial de todas las variables en orden a que el programa no arrastre ninguna condición inicial que pueda entorpecer la ejecución y conducir a un resultado erróneo.

Al introducir el comando RUN desprovisto de argumento, la ejecución empezará a partir de la primera línea del programa. Si fuera necesario empezar la ejecución desde cualquier otra línea distinta de la inicial, habrá que especificar el número de línea en cuestión en la zona de argumento.

El formato genérico de una instrucción RUN es el que sigue:

RUN *<número de línea> <,R>*

Tal como veremos más adelante, con ocasión del estudio de los archivos de información en el BASIC, la opción final "R", permite mantener abiertos todos los ficheros de trabajo que ya se encontraban en esta situación antes de ejecutar el programa.

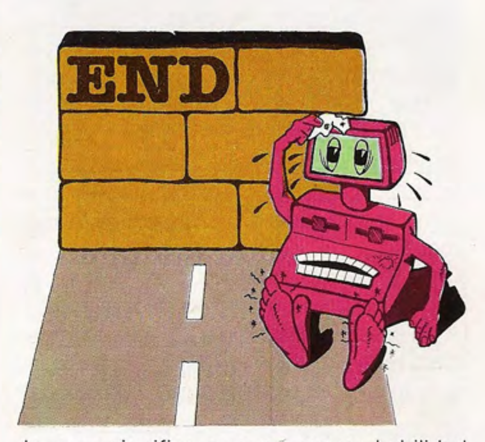
EL COMANDO END

La necesidad de instruir al ordenador precisando cualquier matiz, por obvio que éste sea, es una realidad casi proverbial. Por si fuera preciso corroborarlo, aquí está la instrucción END, cuyo cometido no es otro que advertir al ordenador que ha llegado al final del programa. Por supuesto, no cabe argumento alguno para complementar la actuación de este comando, que por sí solo, constituye una instrucción completa.



El ordenador es una máquina a la que es preciso instruir hasta el más mínimo detalle, hasta el punto de tener que comunicarle donde se encuentra el final de un programa por medio de una instrucción END.

Tras ejecutarla, el ordenador vuelve al «modo comando» (el cursor regresa a la pantalla) y el intérprete BASIC queda dispuesto para seguir prestando su eficaz servicio de intermediario con la interioridad del ordenador.



LOS DATOS DEL BASIC

Sin lugar a dudas la misión primordial del ordenador es el tratamiento de los datos. Pero... ¿Qué tipos de datos? En esencia, un ordenador es una máquina de calcular, lo que significa que su mayor habilidad consiste en trabajar con números. De hecho, sus circuitos más elementales sólo reconocen códigos numéricos (cadenas de ceros y unos). No obstante, un ordena-

END

Instrucción de fin de programa.

Formato: (Número de línea) END

Ejemplos: END

dor dotado de un intérprete BASIC es ya capaz de identificar varios tipos de datos. En el BASIC cabe distinguir dos tipos de datos:

- numéricos y
- alfanuméricos.

Los primeros son ni más ni menos que números convencionales expresados, normalmente, en el sistema decimal. Al segundo tipo pertenece cualquier cadena de caracteres o conjunto de letras, números y caracteres especiales (signos de puntuación, etc.).

Según esta clasificación, un número puede ser considerado como dato numérico o alfabético. Para diferenciar su naturaleza, los números utilizados como datos alfanuméricos suelen ir encerrados entre comillas. Por ejemplo: 1 es un dato numérico, mientras que "1" corresponde a su expresión alfanumérica.

Ambos tipos de datos están presentes en cualquier programa BASIC. La evidencia la encontramos en algunos de los ejemplos propuestos al tratar el comando PRINT.

Al hablar de este comando, se introdujo el concepto de variable o referencia simbólica a la que pueden asignarse datos fijos o constantes. Las variables tienen en el BASIC una función parecida a la propia de las variables matemáticas. Estas últimas se utilizan para designar a un dato desconocido o que puede tomar diferentes valores. En el lenguaje BASIC, las variables tienen un «nombre» que sirve para identificarlas, y un «contenido» o valor que toma la variable en un determinado momento.

Una de las operaciones más frecuentes dentro de un programa es, precisamente, la de alterar el contenido de algunas variables. Este proceso se realiza por medio de denominadas «sentencias de asignación», cuyo cometido es asignar a una variable su correspondiente valor. Este coincidirá con un dato de uno de los dos tipos comentados. Una vez realizada la asignación, el valor o contenido de una variable puede ser utilizado como un simple dato. El comando encargado de la asignación es LET. Su formato es el siguiente:

LET <nombre de variable>=<expresión>

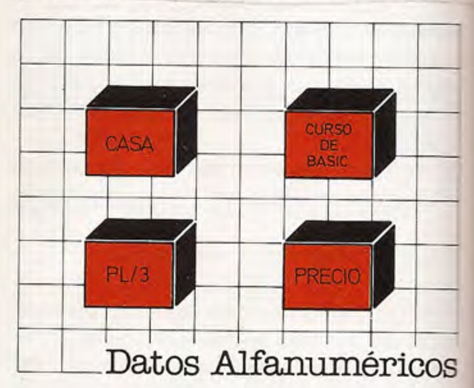
El campo denominado <expresión> puede contener un número, una cadena de caracteres o, en general, una combinación de datos y operadores. Por supuesto, los datos pueden ser constantes o valores numéricos o alfanuméricos o, sencillamente, nombres de variables representativas de su contenido. Las siguientes son asignaciones válidas:

```
LET PI=3.141592
LET NOM$="PACO"
LET SUMA=5+3
LET LONG=2*PI*R
```

En el primer ejemplo se asigna a la variable PI el valor numérico 3.141592. Una vez definida una variable, ésta puede ya utilizarse como dato en otra asignación; la cuarta instrucción LET, utiliza la variable PI (definida en la primera línea) dentro de la expresión cuyo resultado se asigna a la variable LONG. Como se observa, para acceder al valor de una variable basta tan sólo con «llamarla» por su nombre.

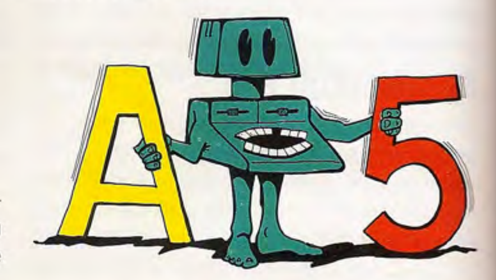
El siguiente programa utiliza tres instrucciones de asignación:

Datos Numéricos



En el lenguaje BASIC coexisten dos tipos de datos: numéricos (números convencionales expresados, normalmente, en el sistema decimal) y alfanuméricos o cadenas de caracteres (conjuntos de letras, números y caracteres especiales).

```
10 LET R=5
20 LET PI=3.141592
30 LET LONG=2*PI*R
40 PRINT "LA LONGITUD ES: "; LONG
50 END
```



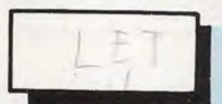
El ejemplo calcula la longitud de la circunferencia cuyo radio (R) se especifica en la línea 10. Para determinar la longitud de una circunferencia de distinto radio, basta con otorgar a R el nuevo valor dentro de la mencionada instrucción.

```
10 LET R=5
20 LET PI=3.141592
30 LET LONG=2*PI*R
40 PRINT "LA LONGITUD ES: "; LONG
50 END
RUN

LA LONGITUD ES: 31.416
```

Cabe observar que los números aparecen en notación inglesa, sustituyendo a la coma decimal por un punto. Esta es una característica casi generalizada en los intérpretes BASIC.

En la mayor parte de los dialectos BASIC,



Asignación de datos o expresiones a variables.

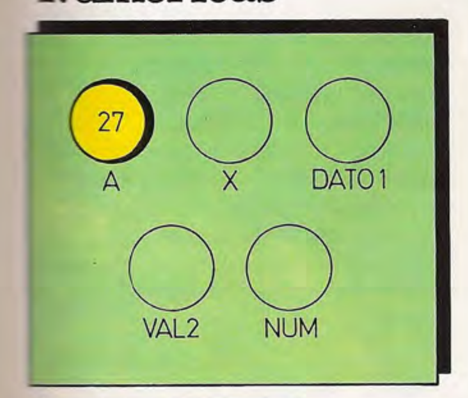
Formato: (Número de línea) LET *<nombre de variable>=<expresión>*

Ejemplos:

```
   LET A=7
10 LET AX$="VARIABLE"
20 LET SUMA=20+250+C
30 LET LONG=2*PI*R
```

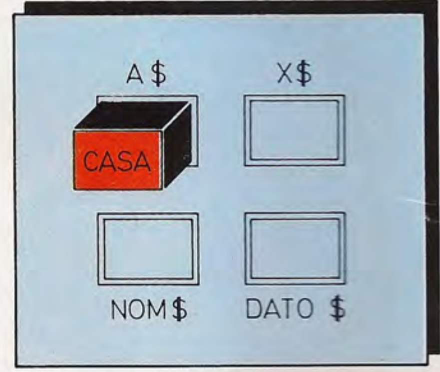
INPUT: Comando destinado a la captación de datos.

Variables Numéricas



LET A = 27

Variables Alfanumèricas



LET A$ = «CASA»

LET es el comando BASIC adecuado para construir las instrucciones de asignación. Su argumento contiene el nombre de la variable y el dato a asignar a la misma: ambos elementos, relacionados por el signo "igual" (=), deben ser del mismo tipo (numéricos o alfanuméricos).

La presencia de la palabra LET es opcional; puede omitirse el comando manteniendo, por supuesto, el formato característico de las instrucciones de asignación. El ejemplo anterior adoptará, en este caso, la forma:

```
10 R=5
20 PI=3.141592
30 LONG=2*PI*R
40 PRINT "LA LONGITUD ES: "; LONG
50 END
```

Ciertas versiones del BASIC admiten la posibilidad de realizar múltiples asignaciones dentro de una misma instrucción LET. El dato o expresión se asigna simultáneamente a todas las variables que lo preceden. Por ejemplo:

```
LET A=B=C=25
```

En esta ocasión, las variables A, B, C tomarán todas ellas el valor 25.

RECOLECTANDO DATOS

En este punto de la obra, el BASIC ha desvelado ya algunos comandos de su repertorio. Comandos útiles para realizar un determinado tratamiento de la información: PRINT (presentar datos en la pantalla), LET (asignar datos a variables)... Se han aportado, incluso, algunos programas sencillos pero ilustrativos de las posibilidades que brinda el BASIC para manipular datos.

Parece obvio que el próximo paso hay que darlo en el sendero de los datos; presentando uno de los comandos BASIC destinado a la captación de datos. INPUT es uno de los comandos de esta categoría. Su especialidad es la de gestionar la entrada de datos durante la ejecución del programa. Al encontrar una instrucción INPUT, el ordenador detiene la secuencia de ejecución y solicita al usuario los datos exigidos por la mencionada instrucción.

La captación de datos a través de INPUT, se reduce a un proceso de asignación. Veamos un ejemplo introductorio:

```
10 PRINT "¿COMO TE LLAMAS?"
20 INPUT A$
30 PRINT "HOLA"; A$
40 END
RUN

¿COMO TE LLAMAS?
?
```

La línea 20 revela uno de los formatos tradicionales de la instrucción INPUT: el comando, seguido de una variable. Como ya se ha mencionado, la introducción de datos se concreta en una asignación. En el ejemplo, el dato que se introduzca quedará asignado a la variable A$, que constituye el argumento; por supuesto, la naturaleza alfanumérica de la variable (A$) obliga a que el dato de entrada sea una cadena de caracteres.

Regresemos de nuevo al ejemplo. Al llegar a la instrucción INPUT, se detiene el proceso de ejecución y aparece un interrogante en la pantalla.

El signo de interrogación que precede al cursor indica, ni más ni menos, que el programa aguarda a que el usuario introduzca un dato. Para que la ejecución pueda continuar, es necesario introducir el dato solicitado, seguido por una acción sobre la tecla RETURN. El efecto de la orden RETURN no es otro que identificar el final del dato.

```
RUN

¿COMO TE LLAMAS?
?MANUEL(RT)

HOLA MANUEL
```

Tras recibir el dato en cuestión, el ordenador vuelve a ocuparse del programa: asigna a A$ el dato "MANUEL" y, por medio de la instrucción 40, lo visualiza en la pantalla precedido del mensaje "HOLA".

Ya se ha mencionado el hecho de que la variable de solicitud de dato (A$ en el ejemplo) puede ser numérica o alfanumérica, exigiendo, en cada caso, un dato del tipo solicitado: número o cadena de caracteres.

Son muchas las posibilidades que este comando pone al alcance del usuario. Una de ellas deriva de la posibilidad de introducir un mensaje en el argumento de INPUT, de tal forma que la solicitud del dato incorpore un texto al efecto.

Veamos un ejemplo. Se trata, sencillamente, de un programa capaz de calcular el precio total de un determinado número de artículos del mismo tipo. El programa incluye las instrucciones INPUT necesarias para pedir al usuario la cantidad de artículos vendidos y el precio por unidad.



El teclado es la principal vía para la entrada de datos al ordenador. Para gestionar su introducción, el BASIC cuenta con instrucciones especializadas; por ejemplo, las que pueden construirse a partir del comando INPUT.

```
10 INPUT "CANTIDAD: "; C
20 INPUT "PRECIO UNITARIO": "; PT
30 PRINT "IMPORTE TOTAL: "; C*PT
40 END
RUN

CANTIDAD: ?5(RT)
PRECIO UNITARIO: ?30 (RT)
IMPORTE TOTAL: 150
```

El signo separador entre el mensaje y la variable que recoge el dato introducido puede ser una coma en lugar de un punto y coma. Esta alternativa suprime la interrogación.

```
10 INPUT "INTRODUZCA SU
NOMBRE: ", A$
20 PRINT "HOLA"; A$
30 END
RUN

INTRODUZCA SU NOMBRE: MIGUEL (RT)
HOLA MIGUEL
```

Otra de las virtudes de INPUT es que tras la introducción de un dato erróneo, no «rompe» la ejecución del programa. En su lugar muestra un *mensaje de error* y, a continuación, vuelve a solicitar el dato.

RUN

PROGRAMA

50 INPUT A

PETICION DE DATO

? A

ENTRADA DE DATO

? A 420

RETURN

Al ejecutar una instrucción INPUT, el ordenador detiene la secuencia de ejecución y solicita al usuario el dato a asignar a la variable que constituye su argumento.

En esta nueva versión del ejemplo inicial, se observa que al utilizar la coma como elemento separador desaparece el signo de interrogación que visualiza el BASIC. Otra particularidad adicional. Si el comando INPUT inmediatamente va seguido por un punto y coma, la orden RETURN que pone fin al dato introducido no provocará un salto a la siguiente línea de impresión. El cursor permanecerá junto al dato ingresado por el usuario. Ello supone que el próximo mensaje se visualizará en la misma línea. Para observar el resultado basta con modificar la línea 10 del ejemplo anterior.

```
10 INPUT; "INTRODUZCA SU NOMBRE", A$
```

El resultado de la nueva ejecución será:

```
RUN

INTRODUZCA SU NOMBRE PABLO. HOLA
PABLO
```

Por último, cabe añadir que una sola instrucción INPUT es perfectamente utilizable para la captación de varios datos, e incluso de distinto tipo. Los datos introducidos se irán asignando, ordenadamente, a las variables incluidas en el argumento. El número de datos introducidos debe ser igual al número de variables que aparecen en la lista y, por supuesto, su naturaleza (dato numérico o alfanumérico) debe coincidir con la de la variable correspondiente. Por ejemplo:

```
INPUT "TRES DATOS"; A,B,C
INPUT "FECHA(AÑO,MES,DIA)";
A,MES$,DIA
```

En el primer caso, hay que responder con tres datos numéricos. No obstante, como respuesta a la segunda instrucción INPUT será preciso introducir un número (A:año), seguido por una cadena de caracteres (MES$), para terminar con un nuevo valor numérico (DIA). Tanto en este caso como en cualquier otra situación análoga, en la

TABLA DE CONVERSION (1)					
ORDENADOR	RUN		END	LET	
	RUN	RUN nl	END	LET <var.>=<expr.>	<var.>=<expr.>
APPLE II (APPLESOFT)	RUN	RUN nl	END	LET <var.>=<expr.>	<var.>=<expr.>
APRICOT (M-BASIC)	RUN	RUN nl	END	LET <var.>=<expr.>	<var.>=<expr.>
ATARI	RUN	RUN nl	END	LET <var.>=<expr.>	<var.>=<expr.>
CBM 64	RUN	RUN nl	END	LET <var.>=<expr.>	<var.>=<expr.>
DRAGON	RUN	RUN nl	END	LET <var.>=<expr.>	<var.>=<expr.>
EQUIPOS MSX	RUN	RUN nl	END	LET <var.>=<expr.>	<var.>=<expr.>
HP-150	RUN	RUN nl	END	LET <var.>=<expr.>	<var.>=<expr.>
IBM PC	RUN	RUN nl	END	LET <var.>=<expr.>	<var.>=<expr.>
MPF	RUN	RUN nl	END	LET <var.>=<expr.>	<var.>=<expr.>
NCR DM-V (MS-BASIC)	RUN	RUN nl	END	LET <var.>=<expr.>	<var.>=<expr.>
NEW BRAIN	RUN	GOTO nl	END	LET <var.>=<expr.>	<var.>=<expr.>
ORIC	RUN	RUN nl	END	LET <var.>=<expr.>	<var.>=<expr.>
SHARP MZ-700 (MZ-BASIC)	RUN	RUN nl	END	LET <var.>=<expr.>	<var.>=<expr.>
SINCLAIR QL	RUN	RUN nl	—	LET <var.>=<expr.>	<var.>=<expr.>
SPECTRAVIDEO	RUN	RUN nl	END	LET <var.>=<expr.>	<var.>=<expr.>
ZX-SPECTRUM	RUN	RUN nl	—	LET <var.>=<expr.>	<var.>=<expr.>

nl: Número de línea. <var.>: Variable. <expr.>: Expresión. Dato o combinación de datos numéricos o alfanuméricos.

FORMULACIONES DE LOS COMANDOS

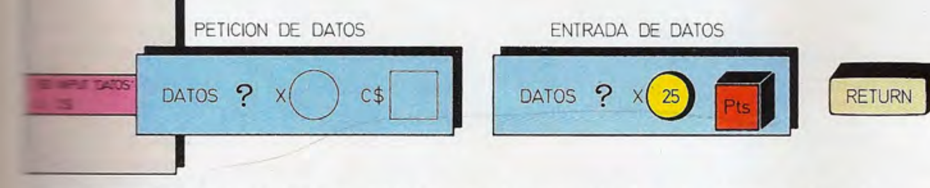
RUN: Ejecuta en su totalidad el programa que se encuentra en la memoria central. RUN nl: Ejecuta el programa a partir de la línea específica (nl). END: Señala el final del programa.

Que haya que introducir varios datos como respuesta a un INPUT, éstos se separarán por medio de una coma.

La versatilidad que permite el comando INPUT puede llegar a simplificar las tareas de programación de forma más que apreciable. Es al usuario a quien corresponde optar por la formulación idónea en cada caso.

Un simple programa, capaz de pedir la introducción de una serie de seis números, operar la suma y presentar el resultado en la pantalla, puede ilustrar la importancia que adquiere la elección del método.

```
20 LET S=0
30 INPUT "A";A
40 LET S=S+A
50 INPUT "B";B
60 LET S=S+B
70 INPUT "C";C
80 LET S=S+C
90 INPUT "D";D
100 LET S=S+D
110 INPUT "E";E
120 LET S=S+E
130 INPUT "F";F
140 LET S=S+F
150 PRINT "SUMA TOTAL=";S
160 END
```



Una misma instrucción INPUT es utilizable para la captación de varios datos, incluso de distinto tipo. Tras introducir los datos, hay que accionar la tecla RETURN para que prosiga la ejecución del programa.

Es evidente que el procedimiento elegido no es el más adecuado. El comando INPUT admite otras formulaciones capaces de reducir la longitud del programa y ha-

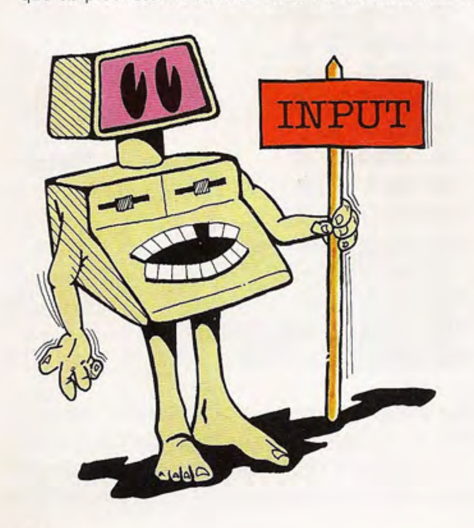
TABLA DE CONVERSION (2)

ORDENADOR	INPUT				
	INPUT <var.>	INPUT "<mens.>";<var.>	INPUT "<mens.>",<var.>	INPUT <var. 1>,<var. 2>...	INPUT;"<mens.>"...
APPLE II (APPLESOFT)	INPUT <var.>	—	—	INPUT <var. 1>,<var. 2>...	—
APRICOT (M-BASIC)	INPUT <var.>	INPUT "<mens.>";<var.>	INPUT "<mens.>",<var.>	INPUT <var. 1>,<var. 2>...	INPUT;"<mens.>"...
ATARI	INPUT <var.>	—	—	INPUT <var. 1>,<var. 2>...	—
CBM 64	INPUT <var.>	INPUT "<mens.>";<var.>	—	INPUT <var. 1>,<var. 2>...	—
DRAGON	INPUT <var.>	INPUT "<mens.>";<var.>	—	INPUT <var. 1>,<var. 2>...	—
EQUIPOS MSX	INPUT <var.>	INPUT "<mens.>";<var.>	INPUT "<mens.>",<var.>	INPUT <var. 1>,<var. 2>...	INPUT;"<mens.>"...
HP-150	INPUT <var.>	INPUT "<mens.>";<var.>	INPUT "<mens.>",<var.>	INPUT <var. 1>,<var. 2>...	INPUT;"<mens.>"...
IBM PC	INPUT <var.>	INPUT "<mens.>";<var.>	INPUT "<mens.>",<var.>	INPUT <var. 1>,<var. 2>...	INPUT;"<mens.>"...
MPF	INPUT <var.>	—	INPUT "<mens.>";<var.>	INPUT <var. 1>,<var. 2>...	—
NCR DM-V (MS-BASIC)	INPUT <var.>	INPUT "<mens.>";<var.>	INPUT "<mens.>",<var.>	INPUT <var. 1>,<var. 2>...	INPUT;"<mens.>"...
NEW BRAIN	INPUT <var.>	—	INPUT ("<mens.>")<var.>	INPUT <var. 1>,<var. 2>...	—
ORIC	INPUT <var.>	INPUT "<mens.>";<var.>	—	INPUT <var. 1>,<var. 2>...	—
SHARP MZ-700 (MZ-BASIC)	INPUT <var.>	INPUT "<mens.>";<var.>	—	INPUT <var. 1>,<var. 2>...	—
SINCLAIR QL	INPUT <var.>	INPUT "<mens.>";<var.>	—	INPUT <var. 1>,<var. 2>...	—
SPECTRAVIDEO	INPUT <var.>	INPUT "<mens.>";<var.>	—	INPUT <var. 1>,<var. 2>...	—
ZX-SPECTRUM	INPUT <var.>	INPUT "<mens.>";<var.>	—	INPUT <var. 1>,<var. 2>...	—

<var.>: Variable. <mens.>: Texto o mensaje.

FORMULACIONES DEL COMANDO INPUT

INPUT <var.>: Entrada de un dato asignándolo a la variable indicada. INPUT "<mens.>";<var.>: Entrada de un dato con presentación del mensaje en la pantalla seguido por el signo de interrogación. INPUT "<mens.>",<var.>: Entrada de un dato con presentación del mensaje omitiendo el signo de interrogación. INPUT <var. 1>,<var. 2>...: Entrada de un conjunto de datos, separados por comas, asignándolos a las variables en el orden en el que éstas aparecen. INPUT;"<mens.>"...: Entrada de datos sin que se produzca un salto de línea tras su introducción.



cer más cómoda la introducción de los datos. Por ejemplo:

```
20 INPUT "A,B,C,D,E,F";A,B,C,D,E,F
30 LET S=A+B+C+D+E+F
40 PRINT "LA SUMA ES: "; S
50 END
```

Ambos programas son de todo punto equivalentes; introduciendo los mismos valores en ambos casos, el resultado será el mismo. Sin embargo, no cabe duda que el procedimiento correcto es el utilizado en el segundo programa.

INPUT

Asigna el valor introducido por el teclado a la variable indicada.

Formato: (Número de línea) INPUT [;] ["*<Mensaje>*" ;,] *<var. 1>*[,*<var. 2>*...]

Ejemplos:

```
INPUT A$
INPUT "VALOR";V
INPUT "DIA",D
```

Logo (2)

El primer contacto con el lenguaje de la tortuga

Sin lugar a dudas, la característica más espectacular del LOGO se encuentra en el TURTLE GRAPHICS. Un método de dibujo que consiste en desplazar una tortuga a través de la pantalla; ésta irá creando el dibujo al dejar visible el rastro de su trayectoria en los sucesivos movimientos. Crear un dibujo se convierte, pues, en algo tan simple y divertido como guiar el desplazamiento de tan simpático personaje.

TEXTO Y DIBUJOS EN LA PANTALLA

Para desarrollar el conjunto de posibles actividades que brinda el LOGO, éste admite tres tipos o formatos de pantalla:

- Pantalla de *texto*
- Pantalla de *gráficos* y
- Pantalla *partida.*

La propia denominación que recibe cada formato de pantalla revela su utilidad: la primera está destinada a la presentación de texto, la segunda a la visualización de gráficos y la tercera, denominada pantalla partida o fraccionada, comparte ambas posibilidades.

En la primera modalidad (pantalla de texto) no es posible ver a la tortuga y tampoco a los dibujos trazados con su colaboración. Este modo de presentación sólo resulta adecuado para la escritura de texto y suele ser el inicial, presente en el instante de conectar el ordenador.

Para acceder a la tortuga es preciso cambiar a otra modalidad de pantalla; para ello, basta con teclear un determinado comando gráfico. El efecto de tal acción será el salto a la pantalla de *gráficos* o a la pantalla *partida.*

Los comandos gráficos que permiten el "salto" a una nueva modalidad de pantalla son: FS, TS y SS.

El primero de ellos (FS: Full Screen) reserva toda la pantalla para gráficos. Si se introduce algún texto después de utilizar el comando FS, el usuario observará que su mensaje no aparece reflejado en ningún punto de la pantalla.

Si el comando introducido es SS (Split Screen), quedará seleccionada la pantalla



El LOGO es uno de los lenguajes más frecuentes en ordenadores de tipo doméstico. Son muchos los modelos para los que existe el adecuado traductor de LOGO, como alternativa al intérprete BASIC habitual.

partida: el texto aparecerá en la parte inferior de la misma, mientras que la zona superior quedará reservada a las evoluciones de la tortuga.
El tercero de los comandos, TS (Text Screen) destina toda la pantalla a la visualización de texto, sin dejar resquicio alguno para la tortuga.
Pasando alternativamente de FS a TS, pueden ir observándose las órdenes introducidas y su ejecución en modo gráfico. Sin embargo, resulta más cómodo el uso de la pantalla partida, tal como tendremos ocasión de estudiar con detalle en los capítulos dedicados a TURTLE GRAPHICS.

COMANDOS Y OPERADORES

En el LOGO es muy importante el concepto de entrada y salida de datos. Imaginemos dos máquinas tragaperras, una de ellas expendedora de chicles y la otra un video-juego de "marcianos". En la primera, al introducir la moneda (dato de entrada), sale de inmediato un paquete de chicles (dato de salida). Por contra, al introducir la moneda (dato de entrada) en la máquina de video-juegos, no cae una nave galáctica por la ranura inferior (no hay dato de salida. Realmente, estamos otorgando la cualidad de "tangibles" a los datos de nuestro ejemplo, con lo cual, parece que no hay dato de salida en el segundo de los casos. Pasemos ahora a la concreción del LOGO. Para empezar, hay que tener en cuenta que las órdenes o instrucciones pueden ser de dos tipos:

- *comandos* u
- *operadores*

(según la terminología propia de este lenguaje).

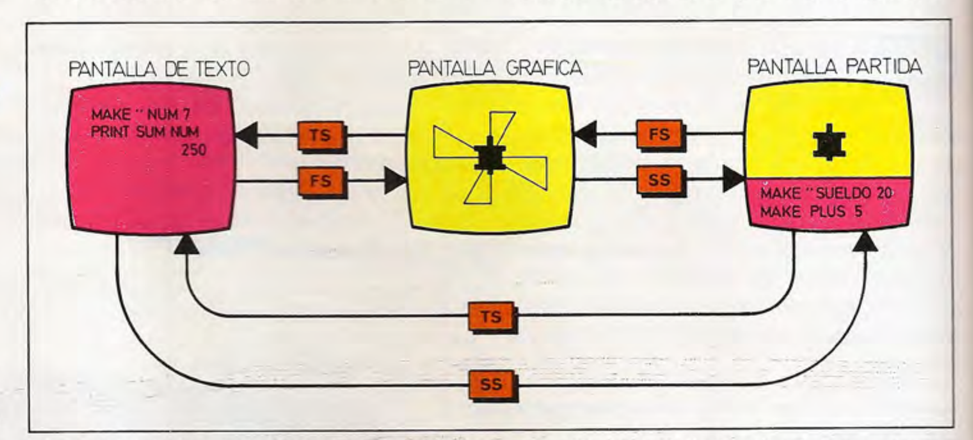
La diferencia entre ambos tipos de instrucciones radica en que un *comando* es una orden que puede o no tener datos de entrada, pero que en ningún caso proporciona dato de salida (el ejemplo de la máquina de video-juegos). Por el contrario, el *operador* es una orden que proporciona siempre algún dato de salida (la máquina expendedora de chicles).
Por ejemplo, la orden FORWARD 5 es un comando: mueve la tortuga cinco espacios hacia adelante (5 es el dato de entrada), pero no proporciona ningún dato de salida. En cambio SUM 2 3, suma los números 2 y 3 entregando el número 5 como resultado. Las cifras 2 y 3 son datos de entrada, mientras que 5 es el dato de salida. Por lo tanto, SUM será un operador.
En el LOGO, las órdenes o instrucciones deben redactarse de tal forma que empiecen por un comando, puesto que los datos de entrada van siempre al final de la instrucción. Esta condición la observamos en cualquier orden, por ejemplo:

FORWARD 5

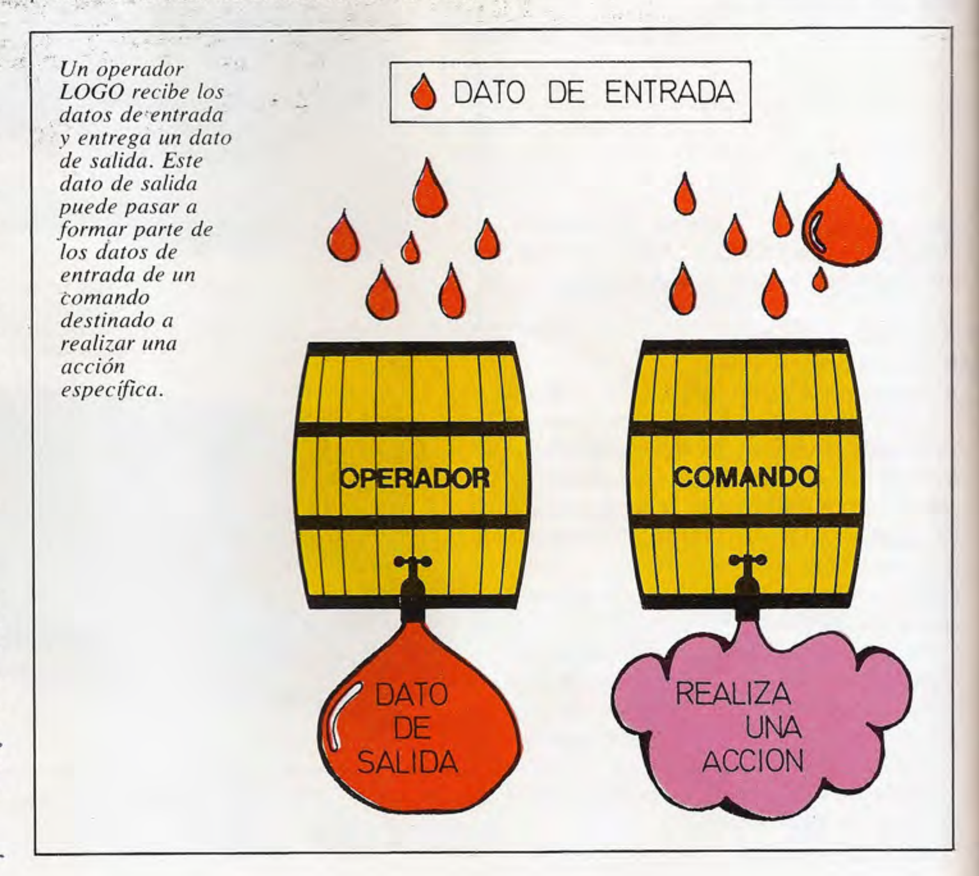
en la que el comando (FORWARD: hacia adelante) precede al dato de entrada (5).
Los datos de entrada pueden coincidir, incluso, con los datos de salida de un operador. En la siguiente instrucción:

FORWARD SUM 2 3

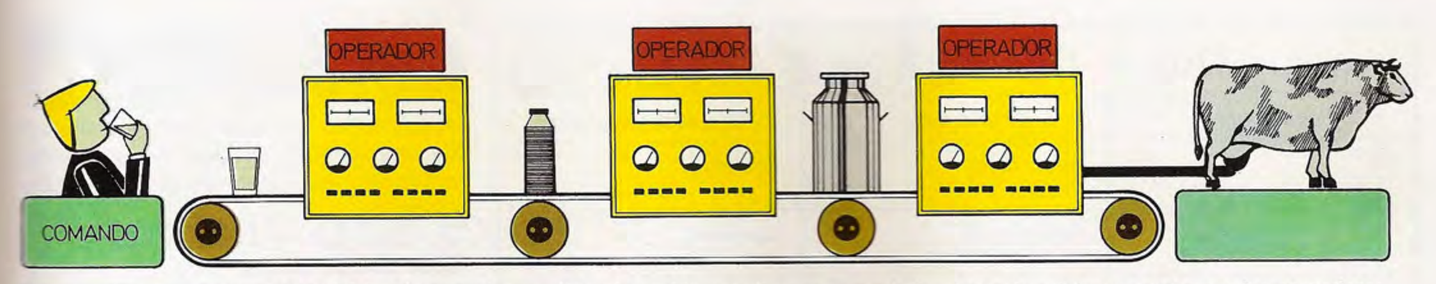
se ordena a la tortuga que avance un número de posiciones coincidente con el resultado de sumar 2 y 3. Por lo tanto, el



Las tres pantallas del LOGO: de texto, gráficos y pantalla "partida". El salto de una a otra se produce al utilizar los comandos TS, FS o SS.



Un operador LOGO recibe los datos de entrada y entrega un dato de salida. Este dato de salida puede pasar a formar parte de los datos de entrada de un comando destinado a realizar una acción específica.



Estructura de una instrucción LOGO. El comando que define la acción a realizar se nutre de los datos de salida generados por los operadores que lo acompañan.

dato de entrada del comando FORWARD es el dato de salida del operador SUM 2 3. A su vez, las entradas de SUM podrían coincidir con los datos de salida de otros operadores. En todo caso, hay que recordar que la primera palabra de la orden o instrucción debe ser un comando; de lo contrario, se perdería el dato de salida generado por el operador que lo sigue. Veamos un ejemplo. La instrucción siguiente:

SUM 284 2000

realiza la suma de los dos números indicados, sin embargo, el resultado no se utiliza para nada; no hay ningún comando al principio de la instrucción que le dé utilidad.

En tal caso, el LOGO mostrará en la pantalla un mensaje de error:

```
SUM 284 2000

YOU DONT SAY WHAT
TO DO WITH 2284
```

(¡No me has dicho qué debo hacer con el dato de salida 2284!).

LAS VARIABLES DEL LOGO

Además de operadores, comandos y datos, existen otros elementos en las instrucciones LOGO: las *variables*.

De forma simple, aunque ilustrativa, puede considerarse a la variable como un compartimento capaz de contener datos. En toda variable hay que distinguir dos partes:

- *nombre de la variable* y
- *contenido.*

El contenido es el valor del dato almace-

Instruyendo a la máquina

El ordenador es una herramienta capaz de demostrar su utilidad y eficacia en múltiples actividades. Para que su capacidad se vuelque en la práctica, es preciso "instruirlo", comunicarle, con toda suerte de detalles, qué debe hacer y cómo debe hacerlo.

Educar a la máquina, *programarla,* supone redactar una completa "receta" de instrucciones; o lo que es lo mismo, confeccionar un programa utilizando el lenguaje propio del ordenador.

Si queremos que un cocinero nos prepare un plato que desconoce, es preciso darle la receta: una serie de instrucciones detalladas que, ejecutadas en su estricto orden, permitirán cocinar el plato. Hay ciertas instrucciones que deben ser especialmente detalladas; por el contrario, otras como "freír" o "pelar", son de sobra conocidas por el cocinero y no exigen mayores precisiones. Hay que tener cuidado para no cometer errores al escribir la receta, puesto que no es lo mismo "pollo frío" que "pollo frito".

Nuestro cocinero es el ordenador, alguien capaz de preparar cualquier plato que se nos antoje, por supuesto, siempre y cuando se le entregue la receta (el programa) adecuado.

Una vez que se conozca con detalle el lenguaje del ordenador, su aplicación a cualquier actividad práctica llegará a constituir una tarea tan cómoda y habitual como realizar la colada. Sabemos cómo comunicarle las instrucciones y cómo aportarle los datos. El se ocupará de procesar los datos de acuerdo a las instrucciones del programa, y entregará el resultado con prontitud y precisión.

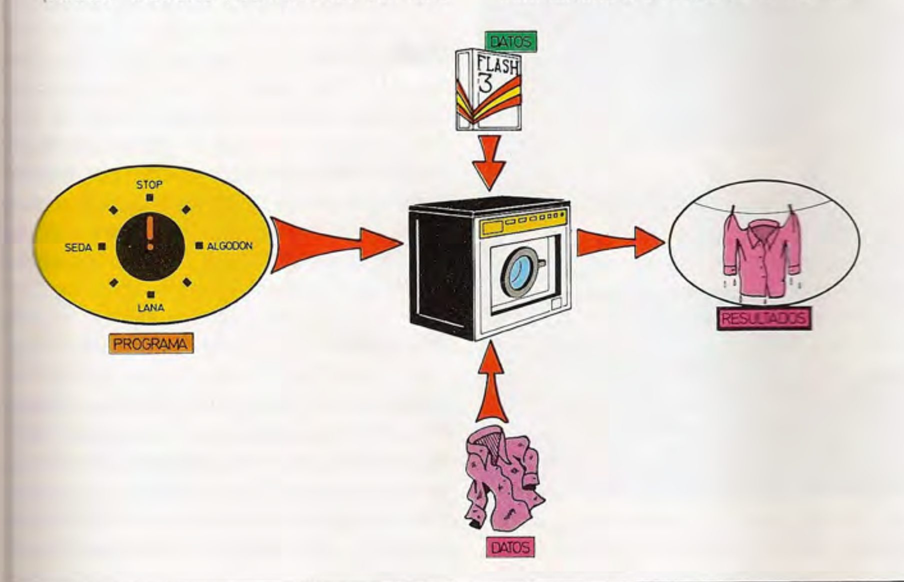
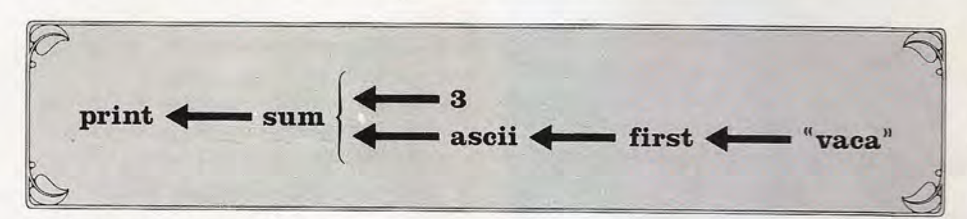
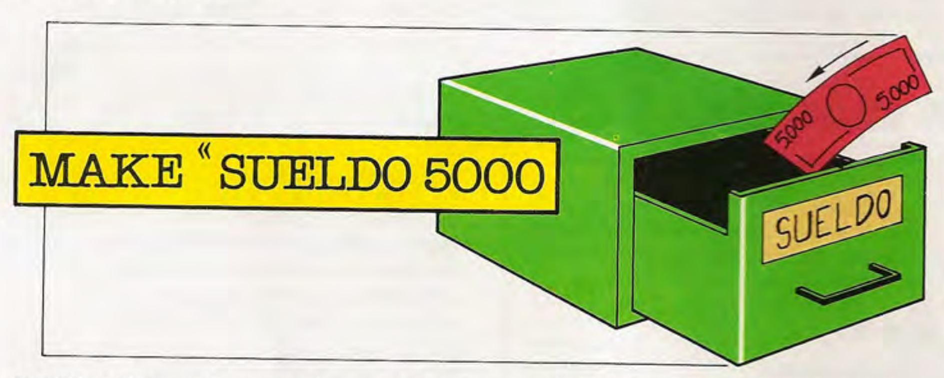


TABLA DE ORDENES-LOGO		
Instrucción	**Cometido**	**Operador/Comando**
FS	Selección de pantalla gráfica	Comando
TS	Selección de pantalla de texto	Comando
SS	Selección de pantalla partida	Comando
MAKE"<palabra> <dato>	Definición de variable	Comando
PRINT <objeto>*	Muestra el <objeto> por pantalla	Comando

* <objeto>: puede ser una palabra, una lista, un número o una variable.



Cualquier instrucción LOGO debe empezar con un comando. Si su lugar estuviera ocupado por un operador, la instrucción no ejecutaría una acción y se perderían los datos de salida de los operadores previos.



MAKE es el comando apropiado para definir una variable LOGO; debe estar seguido por el nombre de variable (precedido por comillas) y por el dato a asignar.

nado en dicho recipiente o *variable*, mientras que el nombre es la palabra que identifica a cada variable y la distingue por completo de las restantes.
Cabe suponer que las variables son algo semejante a los cajones de un armario clasificador. Cada cajón tiene una etiqueta que sirve para identificarlo y, por supuesto, dentro del cajón pueden almacenarse distintos objetos (datos). En el caso que nos ocupa, cada variable tiene un nombre que la identifica con exclusividad y en ella podemos almacenar muy diversos valores numéricos.

Por lo demás, podemos acceder al contenido de un cajón (al dato) a través del nombre que lo identifica (el nombre de la variable). Un buen ejemplo para ilustrar estos conceptos lo encontramos en el fichero de clientes de una empresa. El identificador de cada una de las fichas coincide con el nombre del cliente y, desde luego, es posible consultar una de estas fichas para obtener información. También se puede actualizar su contenido, ya sea borrando, añadiendo o sustituyendo datos. De forma análoga, también es posible consultar, modificar o borrar a voluntad el contenido o dato asignado a una variable.
Los datos de entrada incluidos en una orden, pueden definirse como el contenido de una variable, en lugar de expresarlos en forma de valor numérico fijo. Por ejemplo:

SUM :DATO1 :DATO2

suma el contenido de las variables DATO1 Y DATO2.
Para definir cualquier variable, es preciso utilizar un comando al efecto: MAKE (Hacer).
MAKE admite dos datos de entrada: el primero es el nombre con el que se desea identificar a la variable, mientras que el segundo coincide con el dato que se desea almacenar en la misma.
El nombre se puede elegir libremente, si bien, suele utilizarse una palabra relacionada con la naturaleza del contenido que va a almacenarse en dicha variable. Por ejemplo, si una variable va a contener un número correspondiente al sueldo que percibe un empleado, lo más lógico es otorgar a la variable en cuestión el nombre SUELDO. Hay que precisar, que el nombre de la variable hay que introducirlo precedido por comillas ("). El motivo se comentará en el capítulo dedicado a las "palabras y listas" del LOGO.
El segundo dato que acompaña al comando MAKE debe coincidir con el valor que se desea otorgar a la variable. Más adelante, se verá que el dato en cuestión puede ser de varios tipos, incluyendo la propia salida de un operador o el contenido de otra variable. Por ejemplo:

MAKE "DIA30

asigna a la variable DIA el valor 30.
A la hora de acceder al contenido de la variable, bastará sencillamente con indicar el nombre de la misma, precedido por el signo "dos puntos" (:). Si la instrucción al efecto la empezamos con el comando PRINT:

PRINT :DIA

al ejecutar la orden, accionando la tecla RETURN, el contenido de la variable (DIA en nuestro caso) aparecerá escrito en la pantalla. El signo "dos puntos" que precede a la palabra DIA, indica al LOGO que se trata, precisamente, del nombre de una variable.

Evaluación de un S.O.

¿Qué hay que exigirle a un sistema operativo?

Todo el conjunto de posibilidades del sistema operativo deben tender a un objetivo básico: acercar el ordenador al usuario, poniendo a disposición de éste todos los recursos de máquina. Una labor que obliga al S.O. a aportar las herramientas necesarias para tal fin.

El usuario suele repetir constantemente un determinado tipo de operaciones, con las que es capaz de llevar a cabo sus propósitos. Por este motivo, el sistema operativo incorpora todo un abanico de programas de utilidad capaces de facilitar el manejo de datos y programas. Veamos cuáles son las utilidades más frecuentes.

La instrucción de los datos y del propio texto de los programas fuente en los correspondientes archivos, puede llegar a ser una tarea poco grata si no se dispone de un método eficaz para hacerlo. Con la ayuda de ur editor de textos el trabajo resultará bastante más cómodo al disponer de utilidades que permiten la corrección de errores, la localización de un cierto elemento dentro del conjunto de datos, el cambio de un valor determinado por otro, el borrado y la inserción de nueva información, todo un largo etcétera de funciones que dependerá de la potencia del editor en cuestión.

Uno de los problemas más críticos en todo sistema informático es el de garantizar la integridad de la información puesta en juego, dado el gran volumen que se maneja. La protección y restricción del acceso a los datos deben ser factores contemplados por el sistema operativo, de forma que tan sólo cierto tipo de usuarios puedan acceder y modificar los datos almacenados.

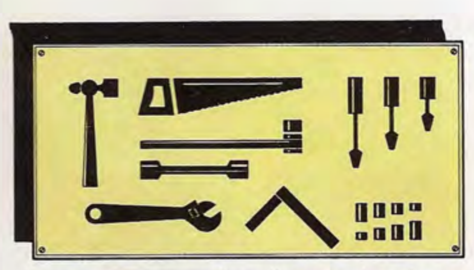
El sistema operativo permite el manejo de la información contenida en los ficheros, brindando al efecto utilidades para facilitar la creación, borrado, copiado y cambio de nombre de los archivos.

AMBITOS DE UTILIZACION DE LOS ORDENADORES

La incorporación del sistema operativo a la máquina, da pie al nacimiento de un ordenador o sistema para el tratamiento de información. En su actuación práctica, éste puede operar en dos ámbitos fundamentales, o lo que es lo mismo, de acuerdo a dos modos básicos de operación:

- *En tiempo real,* o en
- *Explotación secuencial por lotes.*

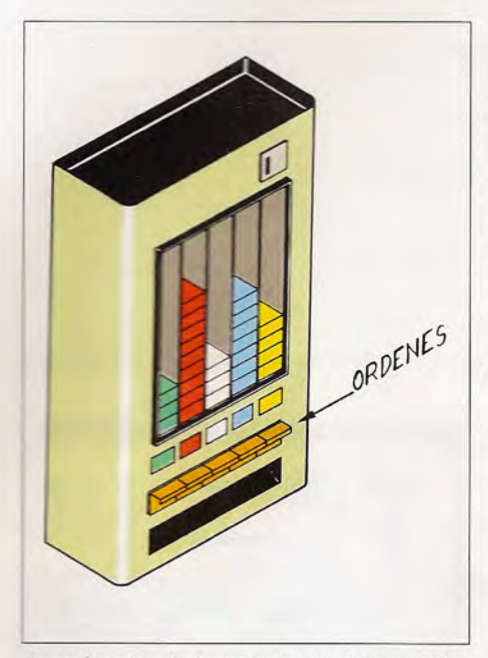
En el funcionamiento en tiempo real, el ordenador es capaz de generar una respuesta inmediata ante una acción o solicitud externa. Por consiguiente, la relación entre el usuario y la máquina es *interactiva.* Este es el método más frecuente en



Los sistemas operativos disponen de todo un repertorio de herramientas de utilidad, destinadas a facilitar al usuario el manejo de datos y programas: utilidades para la edición de órdenes, para la corrección de errores, para la apertura y actualización de archivos...



La existencia de un amplio catálogo de traductores y programas de aplicación, es uno de los factores primordiales que intervienen en la evaluación de un sistema operativo.



La utilización de la máquina en tiempo real permite mantener una comunicación interactiva entre el usuario y el ordenador. Las órdenes introducidas por el usuario son inmediatamente ejecutadas por el ordenador.

el campo de los microordenadores: al introducir una orden, por ejemplo a través del teclado, el ordenador la ejecuta de inmediato, entregando el resultado sin dilación. En la explotación secuencial por lotes, la relación entre el usuario y la máquina pierde su inmediatez; ya no se trabaja en modo interactivo. El ordenador recibe un flujo de trabajos ("JOBs", en terminología inglesa) que irá procesando secuencialmente y sin una especial restricción de tiempo. Este método de explotación tiene su manifestación más importante en el denominado modo *Batch*. En un sistema utilizado en modo Batch, no es precisa la atención del usuario durante la ejecución de los trabajos. Una vez que cada "job" entra en tratamiento, el usuario pierde la posibilidad de intervenir y modificar su desarrollo. Cuando el usuario necesita mantener un diálogo constante con la máquina, por ejemplo, para recibir ayuda del ordenador, debe optar por la actuación en modo interactivo. Por el contrario, si la tarea a ejecutar está perfectamente estandarizada y puede realizarla la máquina sin intervención externa, el usuario puede liberar su atención y optar por el trabajo en modo "Batch".



En la explotación secuencial por lotes, la relación entre el usuario y la máquina deja de ser interactiva. El ordenador recibe un flujo de trabajos que irá procesando secuencialmente, sin exigir la intervención del usuario.

Ambos modos de actuación no se traducen necesariamente en ordenadores distintos. Los sistemas operativos actuales permiten a un mismo ordenador operar en

S.Os. para ordenadores personales

LA FAMILIA CP/M

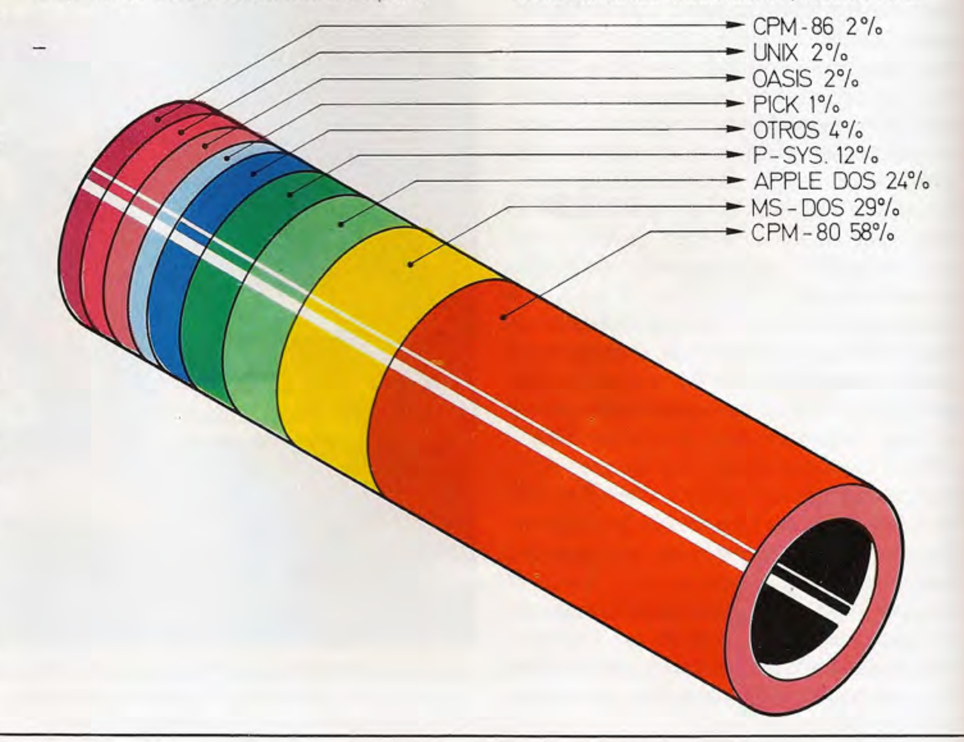
Este fue el primer sistema operativo, de objetivo generalizado, que se desarrolló para su implantación en sistemas basados en microprocesador. Su creador, Gary Kildall, puso a punto una primera versión destinada al microprocesador 8008. Posteriormente se crearon nuevas y cada vez más evolucionadas versiones; las más relevantes son:

- CP/M-80, para los microprocesadores 8080 y Z-80;
- CP/M-86, para el 8086 y 8088;
- Concurrent CP/M, para 8086 y 8088; y
- CP/M-68K, versión destinada al microprocesador 68000 de Motorola.

El CP/M es el primero de los sistemas operativos generalizados que ha saltado del tradicional disco a la memoria ROM. En efecto, una de las últimas versiones es el Personal CP/M, integrado en una memoria de sólo lectura y destinado a ordenadores personales de tipo doméstico.
MP/M-80 y MP/M-86 son versiones para sistemas multiusuario creadas, respectivamente, para equipos de 8 y 16 bits.
La mayor baza de esta popular familia de sistemas operativos reside en la voluminosa biblioteca de programas con que cuenta.

MS/DOS

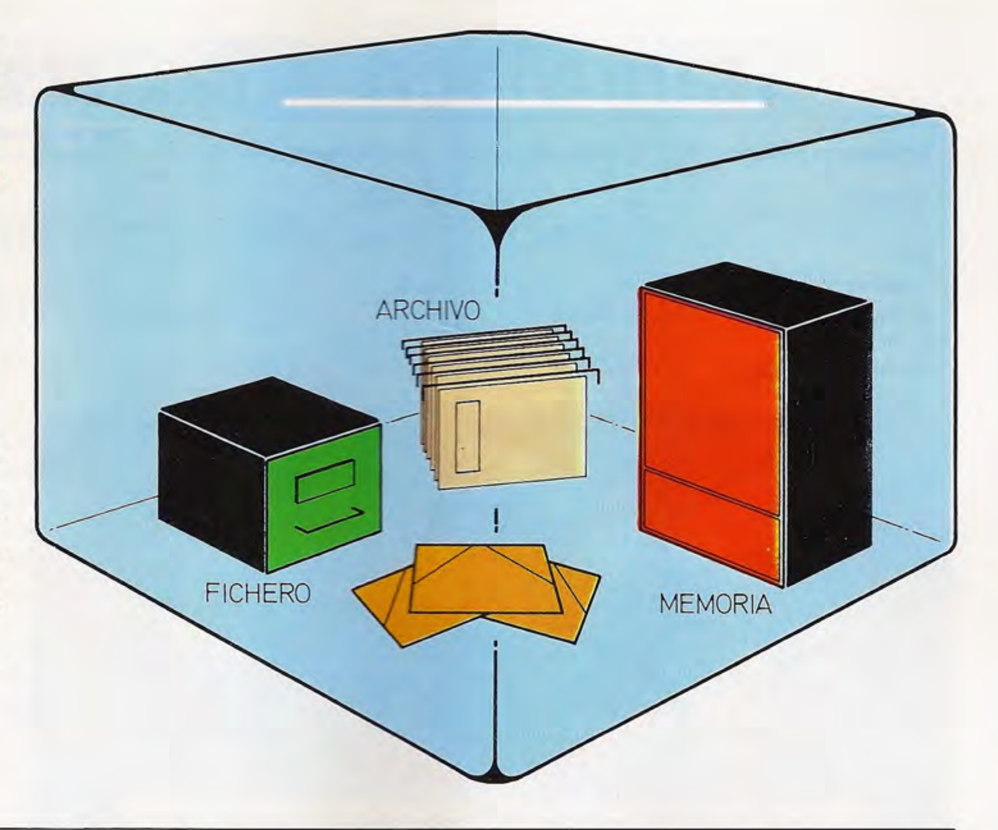
Hace algunos años, la firma Microsoft adquirió los derechos del MS/DOS por la discreta suma de cincuenta mil dólares. Poco más tarde, IBM optó por este sistema operativo para revestir su entrada en el mercado de los ordenadores personales. El



modo interactivo o en modo Batch; la elección, en cada instante, la determina el tipo de tarea, programa o aplicación a procesar.

¿QUE HAY QUE EXIGIRLE A UN S.O.?

Una conclusión evidente es que la misión global del sistema operativo es gestionar la actividad del ordenador. Como tal gestor, a la hora de enjuiciar su trabajo hay que empezar evaluando su eficacia. Esta es una exigencia que no puede determinarse a partir de un criterio único, sino que ➤



Uno de los principales cometidos de cualquier sistema operativo es la protección de los recursos asignados a cada trabajo (zonas de memoria reservadas, archivos de datos y programas...).

impacto del IBM-PC ha convertido al MS/DOS —rebautizado como PC/DOS en el IBM-PC— en un verdadero estándar, llevándolo al liderazgo de los sistemas operativos para microprocesadores de 16 bits. Cada día son más los microordenadores que nacen con el marchamo de "compatibles IBM-PC", y tanto mayor es la difusión del MS/DOS y la amplitud de su biblioteca de programas de aplicación.
Al igual que el CP/M-86, el MS/DOS está concebido para los microprocesadores 8086 y 8088. Las últimas versiones del MS/DOS llegan a incorporar, incluso, un emulador de CP/M-86, lo que permite acceder a los archivos creados bajo el control de este sistema operativo.
La variante multiusuario del MS/DOS es el denominado XENIX; aunque, realmente, este no es más que una versión de otro de los populares: el sistema operativo UNIX. Otra variante del MS/DOS es el MSX/DOS, creado por Microsoft para equipar a los ordenadores domésticos adscritos al estándar MSX.

LA FAMILIA APPLE.DOS

La huella de la firma Apple, uno de los pioneros de la revolución microinformática, sigue aún en plena vigencia. Hasta tal punto que tras el CP/M y el MS/DOS, son los sistemas operativos Apple los que alcanzan una mayor cuota de difusión.
Los equipos Apple II, incluidos los modelos más recientes Apple IIe, Apple IIplus y Apple IIc, incorporan todos ellos sucesivas revisiones del Apple.DOS: el sistema operativo de la propia firma, cuyo producto más relevante es el DOS 3.3.
Otros miembros de la familia son los sistemas operativos SOS —creado para el microprocesador Apple III— y el más reciente ProDOS. Este último, también creado para el microprocesador 6502 y compatible con los anteriores, goza de una notable aceptación entre los profesionales de la programación.
A pesar de la exclusividad de estos sistemas operativos (sólo se encuentran en los microordenadores Apple y en los modelos compatibles de otras firmas), su difusión se mantiene en un nivel elevado. El motivo hay que buscarlo en la gran cantidad de programas de aplicación desarrollados en los últimos años para estos sistemas operativos.
Tema aparte lo constituyen los nuevos y totalmente revolucionarios sistemas integrados, presentes en los modelos Lisa y Macintosh de la propia firma.

UNIX

Algunos sistemas operativos para microordenadores corresponden a migraciones procedentes del terreno de los microordenadores. Tal es el caso del UNIX, un sistema operativo multiusuario que soporta la operación en multitarea.
El mayor número de incondicionales del UNIX cabe localizarlo entre el personal técnico y los programadores profesionales, debido a que su estructura dispone de excelentes útiles para el desarrollo de programas.
El trasvase al campo de la microinformática se ha concretado sobre el microprocesador 68000 de la firma Motorola.
Por el momento, el UNIX parece reservado a los microordenadores más potentes, dotados de una gran capacidad de memoria y con una elevada velocidad de tratamiento.

UCSD p-System

Sus siglas corresponden a la Universidad Californiana de San Diego, en donde se desarrolló la versión original de este sistema operativo monousuario.
La presencia de este sistema operativo es importante en el ámbito de la enseñanza informática. Los principales lenguajes que coexisten con el UCSD p-System, son el Pascal y el Fortran. Una de las versiones del Pascal, desarrollada por la propia Universidad de San Diego (el Pascal-UCSD), encuentra su complemento idóneo en este sistema operativo.

SISTEMAS OPERATIVOS Y MICROPROCESADORES DE ALGUNOS ORDENADORES PERSONALES

Ordenadores personales	Fabricante	País de origen	Microprocesador		Sistemas operativos incorporables al equipo
			Tipo	8/16 bits	
AMSTRAD	Amstrad	Gran Bretaña	Z80A	8	CP/M
APPLE DOS	Apple Computer	EE.UU.	6502	8	ProDOS 1.02, DOS 3.3
CBM 64	Commodore	EE.UU.	6510	8	Commodore DOS
DRAGON 64	Eurohard	España	6809E	8 (1)	OS9
HP 150	Hewlett-Packard	EE.UU.	8088-2	16	MS-DOS 2.11
ICL PC15	ICL	Gran Bretaña	8085A	8	CP/M
IBM PC (y compatibles)	IBM	EE.UU.	8088	16	PC/DOS, MS/DOS, CP/M-86
MACINTOSH	Apple Computer	EE.UU.	68000	16 (2)	Syst. Macintosh, SMALLTALK
MSX	Adscritos al estándar MSX		Z80A	8	MSX/DOS, CP/M
OLIVETTI M20	Olivetti	Italia	Z8001	16	PCOS
PHILIPS P3500	Philips	Países Bajos	Z80A	8	TurboDOS
RANK XEROX 820 II	Rank Xerox	EE.UU.	Z80	8	CP/M
RAINBOW 100	Digital Equip. Corp.	EE.UU.	Z80+8088	8,16	CP/M, MS/DOS
SPECTRUM	Sinclair	Gran Bretaña	Z80A	8	Propio
TOSHIBA T300	Toshiba	Japón	8088	16	MS-DOS, CP/M-86
WANG-PC	Wang	EE.UU.	8086	16	MS-DOS, CP/M-86

(1) El 6809 es un microprocesador con una arquitectura adecuada para el tratamiento de palabras de 16 bits.
(2) Dada su arquitectura interna, el 68000 puede considerarse como un microprocesador capaz de operar tanto con palabras de 16 como de 32 bits.

debe conjugar satisfactoriamente todo un cúmulo de características; algunas dependientes de la propia arquitectura del ordenador que aloja al S.O. Si precisamos nuestra atención sólo en el sistema operativo —omitiendo los condicionantes que impone el hardware de la máquina—, su "eficacia" dependerá de que se aproxime en mayor o menor grado a características como las que enunciamos a continuación:

— Un buen sistema operativo debe aprovechar al máximo las posibilidades hardware del ordenador.

— Durante su actividad debe lograr que la unidad central de proceso opere con un rendimiento máximo. En consecuencia, ésta debe permanecer inactiva el menor tiempo posible.

— Si la actuación es en modo interactivo, la gestión del S.O. debe ser tal que el diálogo ordenador usuario sea rápido: las respuestas deben fluir inmediatamente de la máquina.

— La asignación de recursos debe ser óptima: distribución idónea de la memoria, gestión rápida y eficaz de los periféricos...

— La velocidad de tratamiento debe ser lo más elevada posible; por ejemplo, a la hora de ejecutar trabajos en modo Batch.

— Debe ofrecer al usuario una información completa y detallada del estado del sistema en cada instante: directorios de archivos, mensajes de ayuda...

— Debe proteger los recursos asignados a cada trabajo (zonas de memoria reservadas, archivos...) frente a invasiones de otras tareas o intentos de invasión procedentes de la periferia.

— La versatilidad y eficacia del sistema operativo se ven incrementadas en la medida en que brinda al usuario un mayor repertorio de comandos y funciones.

— La zona de memoria ocupada por el propio sistema operativo debe ser mínima, con objeto de que la mayor parte de los recursos de la máquina queden a disposición de los trabajos a realizar.

Esta relación de criterios de idoneidad puede incrementarse con otras características, complementarias aunque por ello no menos importantes; por ejemplo:

— Es imprescindible que exista un amplio catálogo de traductores de lenguajes y programas de aplicación compatibles con el sistema operativo. Ello permitirá potenciar y dar versatilidad al ordenador.

— Un buen sistema operativo debe estar diseñado para hacer frente a muy diversas contingencias sin perder su eficacia. Si su estructura es modular, la inducción de un error en un módulo no debe tener efecto en los demás.

— Es conveniente que el sistema operativo sea abierto, en orden a facilitar su mantenimiento y actualización.

No cabe duda que la potencia y eficacia del ordenador depende de algo más que de la arquitectura de la máquina. La intervención del sistema operativo es de total importancia a la hora de precisar las posibilidades del ordenador y establecer sus límites de utilidad práctica.



La importancia de los equipos Apple en el mercado actual, tiene su reflejo en el elevado número de usuarios de los sistemas operativos específicos de esta firma.

Software vertical y horizontal

De las aplicaciones específicas, a los paquetes estandarizados

Son varios los criterios que intervienen a la hora de decidir cuál es el ordenador idóneo. No obstante, el camino hacia la decisión parte, habitualmente, del conjunto de tareas que es preciso automatizar.

Una vez precisadas con toda suerte de detalle las características que van a exigirse a cada aplicación, llega el primer interrogante: *¿existe en el mercado algún programa o paquete de programas que satisfaga las condiciones impuestas?*



Los programas que desarrollan un juego forman parte del software vertical. Su actividad es educar al ordenador para que realice una tarea específica.

EXP.
SUMARIO
CLIENTE
ABOGADO
JUEGOS
DEBE HABER
CAJA

SOFTWARE VERTICAL Y HORIZONTAL

La respuesta hay que buscarla en los dos bloques primarios en los que se divide el software de aplicación: software vertical y horizontal.

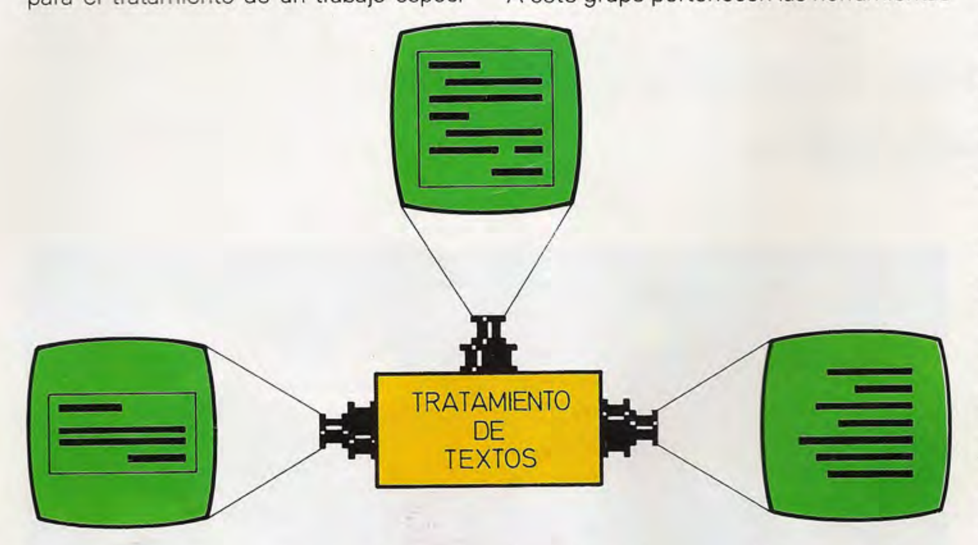
Software vertical

Esta categoría engloba a los programas y paquetes creados para resolver una tarea específica. Por ejemplo, forman parte del



El software vertical acoge a los programas y paquetes creados para resolver una aplicación específica: programas para automatizar el ejercicio de una actividad profesional (médicos, abogados, arquitectos...), o para resolver una tarea especializada (diario de caja, contabilidad general... e incluso juegos).

software vertical los paquetes de aplicación diseñados para automatizar la actividad que rodea al ejercicio de una profesión: médico, abogado, arquitecto... O para el tratamiento de un trabajo específico: confección de la nómina de una empresa, gestión de la actividad de un videoclub, o automatización de la puesta en práctica del Plan General contable en una empresa.



Los programas y paquetes de tipo horizontal resultan adecuados para resolver un gran número de aplicaciones dentro de un determinado marco de actividad.

Software horizontal

El contrapunto del software *específico* o vertical, se encuentra en los paquetes capaces de resolver un gran número de aplicaciones dentro de un determinado marco de actividad. Este tipo de programas y paquetes de aplicación general constituye el denominado software *horizontal.*

A este grupo pertenecen las herramientas de gestión y productividad: tratamientos de textos, hojas electrónicas, programas para la gestión de bases de datos, paquetes para la creación de gráficos, programas para el establecimiento de comunicaciones externas al sistema...

Todos comparten la característica de que el usuario puede orientarlos a cada necesidad específica, dentro del abanico de posibilidades que brinda cada paquete horizontal. Así, por ejemplo, la hoja electrónica es útil tanto al odontólogo, al que facilitará el cálculo de la minuta, como al economista, quien lo utilizará para determinar el precio de venta al público de un producto en función de los costes de producción.

De las cinco categorías en las que suele clasificarse el software de aplicación:

- Juegos/Entretenimiento
- Educación
- Herramientas de gestión y productividad
- Científico-técnicos
- Contabilidad y administración;

el software vertical predomina en todas ellas, excepto en las "herramientas de gestión y productividad", categoría en la que se encuentra el núcleo de los programas y paquetes generalizados u horizontales.

Al margen de los programas estandarizados presentes en los canales comerciales, se encuentra el software confeccionado "a medida", adecuado para satisfacer un trabajo altamente específico. Dada su concreción, es obvio que los programas "a medida" hay que encuadrarlos, normalmente, en el marco del software vertical.

ORDENADORES PERSONALES PARA CUALQUIER APLICACION



En el campo de los ordenadores personales caben muy diversos equipos, de distinta capacidad y potencia, adecuados para ejecutar aplicaciones de mayor o menor complejidad.

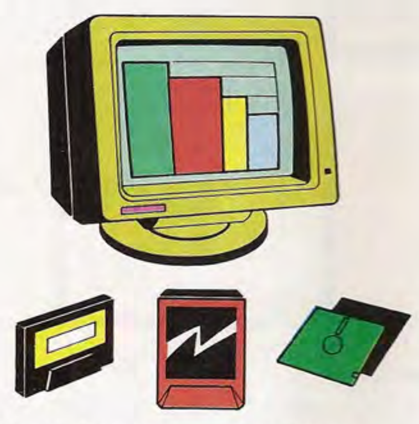
En el mundo de los ordenadores personales caben muy diversos tipos de equipos, desde simples ordenadores de bolsillo (como el popular ZX-81), u ordenadores domésticos (ZX-SPECTRUM, VIC-20, DRAGON, ATARI, COMMODORE 64, ORIC...), hasta potentes ordenadores de gestión (IBM-PC, HP-150, NCR, DM-V...).

De acuerdo con la naturaleza del ordenador personal, éste será capaz de acometer trabajos más o menos complejos. Este es un factor que se manifiesta en el mercado de las aplicaciones. La biblioteca de programas disponibles para ordenadores domésticos se inclina hacia ciertas categorías del software de aplicación; una biblioteca muy distinta de la creada para los ordenadores de tipo profesional o de gestión.

En el caso de los ordenadores domésticos, la mayor profusión de programas corresponde al apartado de juegos y entretenimiento. Aunque, también existen programas, de moderada complejidad, que permiten al usuario editar su correo personal, gestionar la agenda telefónica y manipular los archivos personales, gestionar la contabilidad doméstica o colaborar en labores educativas. Existen incluso algunos programas horizontales que corresponden a versiones simplificadas de los paquetes habituales en los ordenadores personales más evolucionados: hojas electrónicas, tratamien-

Los soportes de memoria del software de aplicación

Al igual que los restantes elementos software del ordenador, los programas y paquetes de aplicación se alojan en las unidades de memoria. Su misión es la de almacenar la información que utiliza la máquina, tanto programas como datos.
En el ordenador cabe distinguir dos tipos de memorias o unidades para el almacenamiento de información: la memoria central o residente en la máquina y las memorias de masa; estas últimas, independientes de la unidad central y asociadas al equipo externamente.
La actividad de cálculo y proceso directo tiene lugar con la información que, en cada instante, reside en la memoria central del ordenador. Si bien, esta zona de almacenamiento interno sólo es capaz de memorizar un volumen de información limitado.
Las unidades de memoria de masa, externas al ordenador, elevan el volumen de información al que puede acceder el ordenador y, en consecuencia, hacen que su actividad de tratamiento pueda alcanzar una mayor potencia y versatilidad. En cualquier caso, no hay que perder de vista que el ordenador sólo puede procesar directamente la información almacenada en su memoria central. Por lo tanto, a la hora de ejecutar un programa o procesar unos datos alojados en la memoria externa, debe empezar trasladando la mencionada información a su memoria central. Una vez realizado el tratamiento oportuno, devolverá la información puesta en juego a su emplazamiento habitual en la memoria de masa.
A la hora de adquirir el software de aplicación, el usuario se encontrará con que éste se encuentra almacenado en un soporte de memoria: cinta de tipo casete, cartucho de memoria ROM o disco flexible. Para que el ordenador pueda ejecutar la aplicación es necesario, pues, ponerla a disposición del mismo, introduciendo el soporte de la aplicación en la correspondiente unidad de memoria que estará asociada a la máquina.
Cada soporte de memoria tiene sus propias características que determinan su idoneidad en cada caso específico. Por ejemplo, la cinta de tipo casete está prácticamente reservada a los ordenadores domésticos. Su reducida capacidad y lentitud, descartan su empleo como memoria de masa de ordenadores de gestión, destinados a tareas complejas. No obstante, su precio moderado convierte a las casetes en soportes adecuados para aplicaciones de juegos, educativas o incluso de gestión a baja escala, destinadas a ordenadores domésticos.



El cartucho de memoria ROM es una alternativa a las cintas de tipo casete en el terreno de los ordenadores domésticos. Aunque es un soporte más caro que la cinta, el acceso a la información almacenada es casi instantáneo, lo que convierte al cartucho en un medio de almacenamiento más práctico. El óbice que impide su uso habitual como soporte para aplicaciones más complejas es su elevado precio y moderada capacidad, además de que sólo permite la lectura de la información almacenada.
El disco flexible es el soporte habitual de las aplicaciones destinadas a ordenadores personales de tipo profesional o de gestión. Los formatos normalizados de los discos flexibles son 8 pulgadas ó 5 y 1/4 pulgadas; sin olvidar a los más recientes micro-floppies, cuyos tamaños más frecuentes son 3 ó 3 y 1/2 pulgadas. Dentro de los discos magnéticos cabe una categoría especial: los discos rígidos. Su uso como memoria de masa es cada vez más habitual en los ordenadores personales, dada su alta capacidad de almacenamiento y velocidad de acceso a la información. No obstante, hay que tener en cuenta que son discos fijos, no extraíbles de la unidad que gestiona su grabación y lectura.





Las herramientas de gestión y productividad dominan el mercado del software horizontal. Su presencia es habitual junto a ordenadores personales de tipo profesional o de gestión.



Las hojas electrónicas y los programas para el tratamiento de textos son dos representantes clásicos popularizados del software horizontal.

tos de textos, gestores de bases de datos o programas para la confección de gráficos.

La mayor potencia y capacidad de la familia de los ordenadores personales se encuentra en los equipos profesionales, portátiles o de sobremesa, y en los orientados a tareas de gestión. El surtido de aplicaciones para estos ordenadores se decanta hacia los paquetes verticales, destinados a profesionales o a pequeñas y medianas empresas, y hacia las herramientas de gestión y productividad. Esta última es la categoría del software horizontal que goza de un repertorio más dilatado de programas.

HERRAMIENTAS DE GESTION Y PRODUCTIVIDAD

El mercado del software horizontal está dominado por un conjunto de paquetes de aplicación, destinados a actividades de gestión y productividad. Paquetes de programas capaces de prestar un eficaz servicio en un dilatado abanico de situaciones.

A esta gama de herramientas software se debe, en gran parte, el vertiginoso empuje comercial de los paquetes estandarizados. De su importancia aboga el hecho de que la adquisición de un ordenador personal, ya sea para uso profesional o para la gestión en el ámbito de la empresa, suele simultanearse con la de algún paquete de esta categoría.

El conjunto de herramientas de gestión y productividad se divide en varios grupos o áreas de actividad:

- Tratamiento de textos
- Hojas electrónicas
- Gestión de archivos y bases de datos
- Paquetes gráficos
- Paquetes de comunicación.

Cinco grupos básicos que pueden completarse con otros de menor relevancia como, por ejemplo, los paquetes para la generación de programas, o los modernos gestionadores de tiempo, ideas, tareas y proyectos.

El predominio se encuentra, sin lugar a dudas, en los cinco primeros grupos. Dentro de cada uno de ellos se encuentran paquetes de programas cuya denominación (Wordstar, Multiplan, Visicalc, dBASE II...) tiene mayor notoriedad en el mundo informático que el propio nombre de muchos ordenadores.

Operando con el BASIC

LIST y REM: dos comandos para auxiliar al programador. Los operadores aritméticos elementales

Los ordenadores son, en esencia, máquinas concebidas para almacenar y procesar información. El concepto de información relativo a los ordenadores tiene algo que ver con la aceptación coloquial de este término.

En general, información es todo aquello que incrementa nuestro conocimiento. Trasladándolo al caso del ordenador, el concepto de información es aplicable a todo aquel material que se suministra a la máquina, ya sea para instruirla (comandos, órdenes, instrucciones, programas) o para que ésta lo opere y manipule de acuerdo a las indicaciones que reciba.

Es evidente que la distinción señalada equivale a dividir la información en dos categorías: los datos, que pueden ser números, letras, palabras..., y los programas o conjuntos de instrucciones que indican al ordenador la tarea a realizar.

Toda esta información ha de quedar al alcance de la máquina para que le sea posible utilizarla en el momento adecuado. Del almacenamiento de la misma se ocupan las unidades de memoria, ya sean residentes en el corazón del ordenador y a disposición directa e inmediata de la unidad central de proceso, o asociadas como dispositivos externos a la máquina.

La confección de programas —información destinada a "instruir a la máquina"— es, precisamente, el objetivo práctico de los lenguajes de programación. Estos deben aportar el vocabulario adecuado para expresar cualquiera de las acciones habituales; además, deben ofrecer al usuario un surtido de órdenes que faciliten la confección de programas. Dentro de este último grupo se encuentran dos comandos BASIC que se van a exponer a continuación: LIST y REM.

EL COMANDO LIST

Una necesidad obvia del programador, es la de ver en cualquier momento el resultado de su trabajo. En definitiva, obtener en la pantalla una lista ordenada de las instrucciones que ha ingresado en el ordenador para construir un programa.

Tal posibilidad la ofrece el comando LIST. Este permite *visualizar* (LISTar) el programa almacenado en memoria en ese preciso instante.

Pero no radica ahí la solución de todo el problema... ¿Qué sucede cuando el programa es suficientemente grande como para no poder visualizarlo en el espacio que ofrece una sola pantalla? Para solucionar este inconveniente, el comando LIST ofrece toda una serie de opciones que permiten el cómodo examen del programa a través de la pantalla.

Las instrucciones LIST, cuyo formato general es:

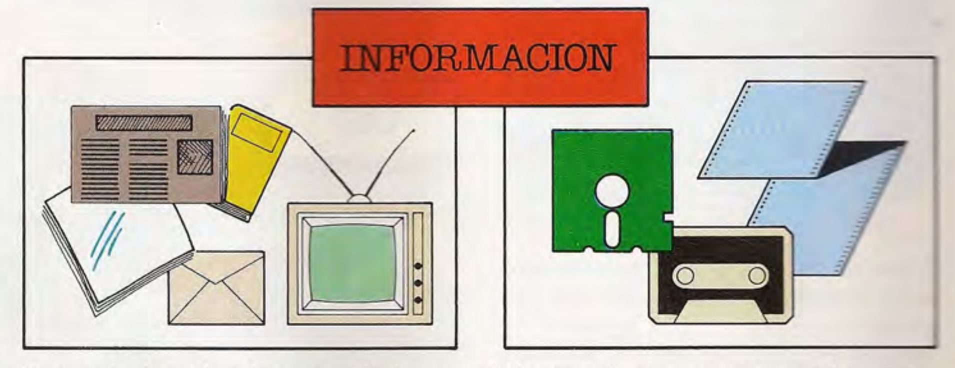
LIST [<Número de línea> [–[<Número de línea>]]]

permiten "listar", total o parcialmente, el programa que se encuentra almacenado en la memoria del ordenador.

Las distintas opciones que ofrece el comando, pueden ser seleccionadas mediante la inclusión o no de los campos opcionales a los que hace referencia el formato general.

Para estudiar las distintas modalidades de instrucciones que pueden construirse con el comando LIST, se utilizará un mismo programa ejemplo, cuyo listado completo es el que aparece a continuación:

```
10 PRINT "INSTRUCCIONES"
20 PRINT "LIST:"
30 PRINT "ESTO NO ES ";
40 PRINT "MAS QUE UN"
50 PRINT "EJEMPLO"
60 END
```



En esencia, los ordenadores son máquinas concebidas para almacenar y procesar información. En el caso del ordenador, el concepto de información es aplicable a todo aquel material que se suministra a la máquina, ya sea para instruirla (órdenes, instrucciones, programas) o para que proceda a su tratamiento (datos).



El objetivo de los lenguajes de programación es construir programas: secuencias ordenadas de instrucciones capaces de «educar» al ordenador para que realice un determinado trabajo.

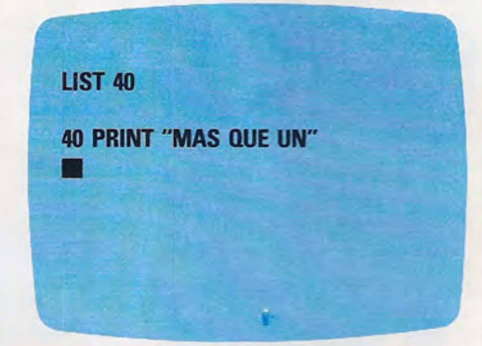
La primera de las instrucciones a la que da pie el uso de este comando consta, sencillamente, del comando LIST aislado, omitiendo la zona de argumento. La respuesta del ordenador será la presentación del programa completo, listado a partir del número de línea inferior.
En el caso de que el programa que se encuentra en la memoria sea el propuesto en el párrafo anterior, el efecto de la orden LIST será el siguiente:

```
LIST

10 PRINT "INSTRUCCIONES"
20 PRINT "LIST:"
30 PRINT "ESTO NO ES ";
40 PRINT "MAS QUE UN"
50 PRINT "EJEMPLO"
60 END
■
```

Como se observa, el ordenador muestra todas y cada una de las líneas del programa.
También es posible "listar" tan sólo parte del programa. Para ello, hay que especificar en el argumento de LIST la línea o las líneas deseadas.

Una primera alternativa es introducir como argumento un solo número de línea. La respuesta de la máquina será la visualización exclusiva de tal línea; por ejemplo:



Si el número de línea va seguido por un guión, se listará la mencionada línea y todas las restantes que tengan un número superior al indicado. Esto es: se mostrará en la pantalla la zona del programa que va desde la línea señalada tras el comando LIST hasta el final del mismo:

```
LIST 30–

30 PRINT "ESTO NO ES ";
40 PRINT "MAS QUE UN"
50 PRINT "EJEMPLO"
60 END
■
```

Por el contrario, si el guión precede al número de líneas especificado, se listará el programa desde el principio hasta llegar a ese número de línea. Por ejemplo:

```
LIST –30

10 PRINT "INSTRUCCIONES"
20 PRINT "LIST:"
30 PRINT "ESTO NO ES ";
■
```

Una nueva variante de la instrucción LIST es la que incluye en el argumento dos números de línea, separados por un guión. En tal caso, se listarán todas las líneas del programa cuyo número esté comprendido entre ambas. Esta última opción resulta útil para presentar en pan-

LIST

Presenta en pantalla las líneas del programa almacenado en memoria.

Formato: LIST[<*número de línea*>[–[<*número de línea*>]]]

Ejemplos: LIST
LIST 3–100
LIST –200
LIST 50–

...talla sólo determinados bloques o zonas del programa en curso.

```
LIST 20—40

20 PRINT "LIST:"
30 PRINT "ESTO NO ES ";
40 PRINT "MAS QUE UN"
■
```

Hay que indicar que este comando permite listar cualquier programa almacenado en la memoria del ordenador. Si bien, es preciso que el programa en cuestión esté escrito en BASIC. Esta salvedad se refiere al hecho de que LIST no resulta adecuado para listar un programa escrito en *código máquina,* ni desde luego un escrito en cualquier otro lenguaje. Este último caso es manifiestamente imposible, ya que, probablemente, la máquina ni tan siquiera reconocerá el comando.
Tras la ejecución del comando LIST, el BASIC regresa al modo directo. Ello significa que si se introduce tal comando en

list



¿Qué es y qué no es un ordenador personal?

Cualquier ordenador, sea cual fuere su tamaño y potencia, es un producto de la síntesis de dos elementos complementarios: un soporte físico o circuito electrónico, el *«hardware»,* y una programación o conjunto de instrucciones, datos, programas..., el *«software».* Ambos elementos se conjugan en un sistema para el tratamiento de información u ordenador. Aquí aparece la distinción esencial entre el ordenador y otras máquinas capaces sólo de resolver un determinado número de tareas específicas (una calculadora, por ejemplo). El ordenador es un sistema cuya funcionalidad no está predefinida por su estructura física, sino que puede ser «instruido» por el usuario para realizar una u otra función introduciéndole un programa al efecto.
Una calculadora convencional es capaz de realizar ciertas operaciones matemáticas (suma, resta, multiplicación, división...), pero única y exclusivamente esas operaciones preestablecidas. Por su parte, el ordenador posee un campo de aplicación totalmente versátil, definible en cualquier instante por medio del adecuado programa. Puede realizar cálculos complejos en los que intervengan secuencias de operaciones, comparaciones y decisiones, y su efectividad no se limita a los cálculos matemáticos, sino que se extiende a cualquier aplicación definible como tratamiento de información sea ésta de naturaleza simplemente numérica o alfanumérica.
La diferencia del ordenador con otras máquinas más sofisticadas y que, en muchos casos, incorporan en su interior un microprocesador, es ya más sutil. Las consolas de video-juegos constituyen un perfecto ejemplo de equipo dotado de una unidad central de proceso integrada (un microprocesador), y capaz de ejecutar un programa (el cartucho de juego). En este caso, la distinción básica reside en que tales máquinas siguen encerradas dentro de un marco de aplicación específico: la ejecución de juegos sobre una pantalla. No están abiertas a lenguajes de programación que versatilicen sus posibilidades, y tampoco disponen de un sistema operativo que ponga toda su potencialidad en manos del usuario.
Una vez delimitado el terreno de los ordenadores o máquinas programables para el tratamiento de la información, llega el momento de caracterizar al ordenador personal.
No hay que olvidar a los grandes ordenadores y tampoco a los miniordenadores, e incluso a los microordenadores evolucionados. Aunque cada vez son mayores las intersecciones entre las diversas categorías de ordenadores, puede enmarcarse al ordenador personal en una zona propia. Dentro de un ámbito delimitado por su definición más amplia: máquina programable basada en microprocesador, destinada al tratamiento de información y orientada al usuario individual; con una gama de periféricos, sistemas operativos, lenguajes de programación y programas de aplicación concebidos específicamente para su explotación.

CALCULADORA

ORDENADOR PERSONAL

VIDEOJUEGOS

SUPER-ORDENADOR



Una de las características propias de los lenguajes de programación es la de brindar al programador herramientas para facilitar su trabajo. LIST y REM son dos de los comandos que ofrece el BASIC para este cometido.

forma de instrucción indirecta dentro de un programa, la *ejecución* del mismo se detendrá tras listarlo. Un ejemplo de la actuación del comando LIST utilizado a modo de instrucción indirecta, lo aporta el siguiente programa:

```
10 PRINT "LIMITACIONES DEL LIST"
20 PRINT "ESTA LINEA SI SE EJECUTA"
30 LIST
40 PRINT "ESTA LINEA NUNCA SE
   EJECUTARA"
50 END
```

En efecto, la ejecución se detiene al procesar la instrucción LIST: el ordenador presenta el listado del programa y, acto seguido, abandona la secuencia de ejecución para pasar a modo directo o situación de diálogo:

```
RUN
LIMITACIONES DEL LIST
ESTA LINEA SI SE EJECUTA
10 PRINT "LIMITACIONES DEL LIST"
20 PRINT "ESTA LINEA SI SE EJECUTA"
30 LIST
40 PRINT "ESTA LINEA NUNCA SE
   EJECUTARA"
50 END
```



La actividad del ordenador

¿Para qué sirve un ordenador? Este es uno de los primeros interrogantes que surgen al tomar un primer contacto con el mundo de la informática. La respuesta no deja de resultar problemática por su difícil concreción. El motivo radica en que un ordenador puede hacer, o por lo menos ayudar a hacer, casi todo; siempre que cuente con el programa adecuado y con el apoyo de los periféricos idóneos.

A la hora de adquirir una idea real de lo que cabe esperar de un ordenador, es muy conveniente empezar conociendo cómo trabaja. Para ello, resulta ilustrativo comparar la actividad de la máquina con el trabajo cotidiano en la oficina de facturación de una empresa.

La oficina está situada en una habitación provista de dos ventanillas, una de recepción o entrada y otra de salida. El empleado necesita tener a mano el conjunto de normas que detallan su trabajo, así como los datos necesarios para realizar cada tarea. Para ello dispone de un tablero, donde coloca el albarán que contiene los datos de entrada. Este albarán incluye el nombre del cliente con el que se efectúa la operación, así como el tipo de artículos y la cantidad de los mismos.

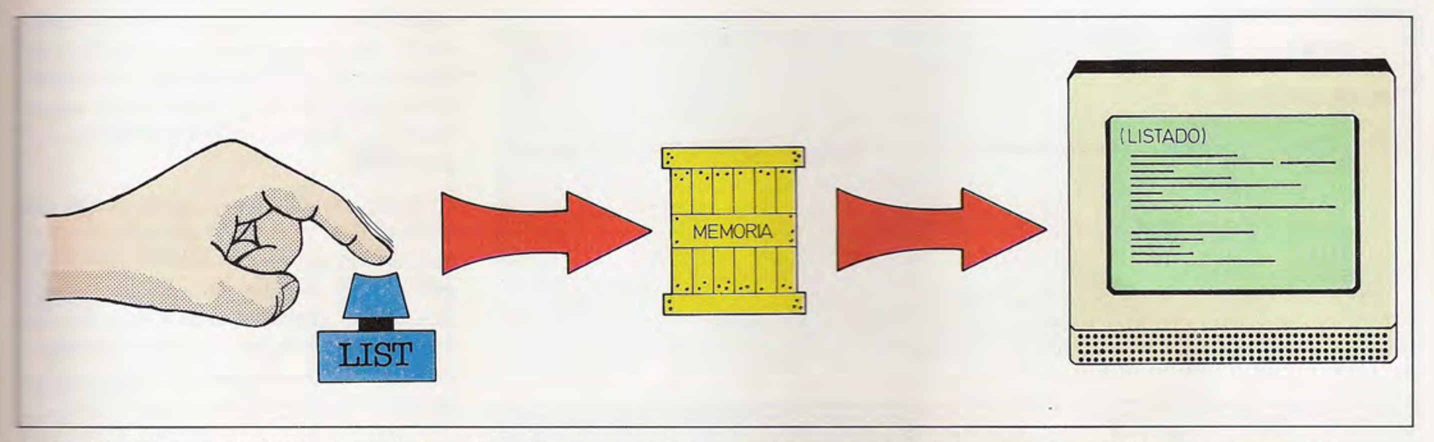
Sobre el tablero también hay un folleto con la tarifa de precios vigente y con información sobre los descuentos a aplicar a cada cliente. Al lado de estos datos se encuentra la lista de normas a seguir para realizar el trabajo. Algo semejante a un *programa* cuyas instrucciones detallan, paso a paso, las operaciones a poner en práctica. Como herramienta de cálculo, el administrativo dispone de una pequeña calculadora.

Dado el volumen de la información puesta en juego, es posible que el espacio disponible en el tablero sea insuficiente. Por esta razón el operario dispone de un archivo localizado en una habitación contigua. En este almacén residen los ficheros que guardan la información que no es posible mantener en el tablero por falta de espacio.

En resumidas cuentas, el administrativo lleva al tablero, exclusivamente, la información necesaria para ejecutar una parte del trabajo. Cuando necesita más datos, por haber concluido esa parte del trabajo o porque los datos no son ya los adecuados, actualizará la información adherida al panel. Esta es una situación parangonable con la actividad y con el método de trabajo habitual en un ordenador. Cada uno de los elementos descritos tienen su homólogo en la máquina.

La *ventanilla de recepción* equivale al *órgano para entrada de información* (normalmente, el teclado) del ordenador. El *administrativo,* rodeado de sus herramientas operativas (papel, lápiz, calculadora...) gestiona el tratamiento de la información; tarea ésta encomendada a la *unidad central de proceso* del ordenador (el microprocesador, en el caso de un ordenador personal).

El *panel* tiene su reflejo en la *memoria central* del ordenador, en la que se



La función del comando LIST, en sus distintas formulaciones, es presentar en la pantalla un listado total o parcial del programa en curso almacenado en la memoria del ordenador.

Al recibir la orden RUN, el ordenador ha iniciado la ejecución del programa, cursando la tarea ordenada en las instrucciones 10 y 20. Al llegar a la línea 30, la máquina ejecuta la instrucción LIST, mostrando en pantalla el listado completo del programa... y, acto seguido, se detiene mostrando el cursor. Las líneas 40 y 50 quedan sin ejecutar, debido a que la instrucción LIST (línea 30) obliga al intérprete BASIC a abandonar la ejecución en curso y a regresar a modo directo.

EL COMANDO REM

A primera vista, la presencia de este comando dentro del vocabulario de un lenguaje de programación, puede parecer un despropósito. REM es un comando que no tiene efecto alguno en la ejecución de un programa BASIC, hasta el punto de que es ignorado por el ordenador. No obstante, en determinados casos, su presencia dentro de un programa llega a ser casi esencial. La misión del REM (del inglés: REMARK, Comentario) es introducir comentarios en el programa que faciliten su interpretación posterior por parte del propio programador o de otros usuarios.. En programas cortos su empleo es casi superfluo, ya que con un simple vistazo se puede saber cuál es el cometido del pro-

almacenan los datos y programas en curso de ejecución. Este cuenta con el auxilio de un *archivo* de gran capacidad, al que se traslada o del que se extrae la información de trabajo: la *memoria de masa* del ordenador.
Por último, el resultado del trabajo acometido, la factura dirigida al cliente, abandona el recinto a través de la *ventanilla de salida*. Una imagen paralela al *órgano de salida* del ordenador (por ejemplo, la pantalla de visualización).

VENTANILLA DE ENTRADA

OFICINA

VENTANILLA DE SALIDA

ENTRADA DE INFORMACION

MEMORIA DE MASA

MEMORIA RESIDENTE

ORDENADOR

C.P.U.

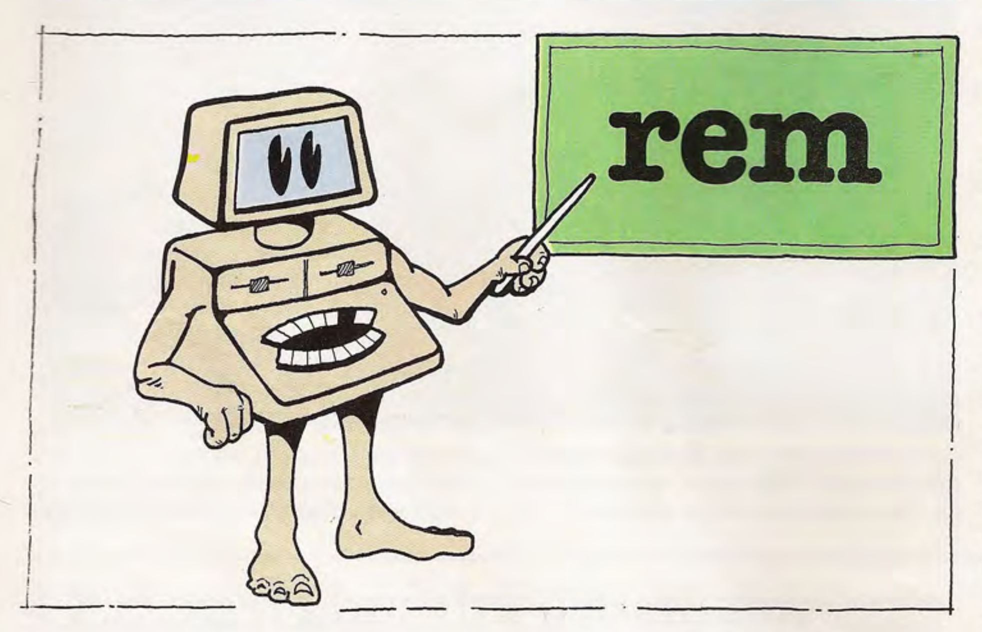
SALIDA DE INFORMACION

REM

Introduce un comentario en el programa. Este comando y el texto que le sigue no se ejecutan.

Formato: (Número de línea) REM [<comentario>]

Ejemplos: REM
REM ESTO ES UN COMENTARIO
REM PROGRAMA 56-B



grama y la función de las variables empleadas. No obstante, cuando el programa sea un poco largo y complicado, su presencia resultará providencial. Si no se introducen comentarios explicando la función de cada parte del programa y el cometido de las distintas variables, ni al mismo programador que lo ha diseñado le resultará fácil revisar su estructura e introducir nuevas modificaciones en tal programa, transcurrido un cierto tiempo desde su confección.

```
10 REM PRINCIPIO DE LA ZONA DE CALCULO
20 REM PROGRAMA REVISADO EL 10 DE
OCTUBRE
30 REM LA VARIABLE P CONTIENE EL PRECIO
EN DOLARES
```

Como se observa en el ejemplo, el formato del comando REM es de lo más simple. Basta con empezar la instrucción con la palabra REM y añadir a continuación el texto del comentario.

En algunas versiones del lenguaje BASIC, es posible sustituir la palabra clave REM, por un simple asterisco (*), por un apóstrofe ('), por un signo de admiración (!) o por cualquier otro símbolo específico.

OPERADORES ARITMETICOS

Sin lugar a dudas, un ordenador desde luego es mucho más potente que una calculadora. En principio, puede compartir todas las posibilidades de una calculadora: desde las operaciones aritméticas elementales hasta cálculos en los que intervengan funciones trigonométricas y algebraicas.

A pesar de ello, hay que señalar que el modo de operación no acostumbra a ser tan inmediato como el propio de una calculadora. No hay que perder de vista que, en el caso del ordenador, hay que ajustarse a las reglas de sintaxis y ortografía propias del lenguaje de programación utilizado; del BASIC en nuestro caso.

Las operaciones matemáticas básicas realizadas por un ordenador, instruido con el lenguaje BASIC, coinciden con las habituales en una calculadora: suma, resta, multiplicación, división y potenciación.

Los signos u *operadores* asociados a cada una de las citadas operaciones son los que muestran la tabla adjunta.

En el lenguaje BASIC, los operadores aritméticos se utilizan para establecer relaciones matemáticas entre los datos, dentro del argumento de las instrucciones.

Este cometido es extensivo a los dos modos básicos de formulación de las instrucciones: directo (sin número de línea) e indirecto (con número de línea, formando parte de un programa). Naturalmente, utilizando instrucciones directas, que obtienen del ordenador una respuesta inmediata, puede simularse el funcionamiento propio de una calculadora. Por ejemplo:

```
PRINT 2+4 (RT)
6
PRINT 5-6 (RT)
-1
PRINT 3*8 (RT)
24
■
```

Operación	Comentario	Ejemplo	Resultado
+	suma	6+4	10
−	resta	3−8	−5
*	multiplicación	5*7	35
/	división	8/4	2
∧ ó ↑	potenciación	6↑3	216

TABLA DE CONVERSION							
ORDENADOR	LIST					REM	
	LIST	LIST nl	LIST −nl	LIST nl−	LIST nl1−nl2	REM	Signo equivalente
APPLE II (APPLESOFT)	LIST	LIST nl	LIST ,nl	LIST nl,	LIST nl1,nl2	REM	—
APRICOT (M9BASIC)	LIST	LIST nl	LIST −nl	LIST nl−	LIST nl1−nl2	REM	'
ATARI	LIST	LIST nl	LIST −nl	LIST nl,	LIST nl1,nl2	REM	—
CBM 64	LIST	LIST nl	LIST −nl	LIST nl−	LIST nl1−nl2	REM	—
DRAGON	LIST	—	LIST −nl	LIST nl−	LIST nl1−nl2	REM	'
EQUIPOS MSX	LIST	LIST nl	LIST −nl	LIST nl−	LIST nl1−nl2	REM	—
HP-150	LIST	LIST nl	LIST −nl	LIST nl−	LIST nl1−nl2	REM	Signo equivalente
IBM PC	LIST	LIST nl	LIST −nl	LIST nl−	LIST nl1−nl2	REM	Signo equivalente
MPF	LIST	LIST nl	LIST −nl	LIST nl,	LIST nl1,nl2	REM	—
NCR DM-V (MS-BASIC)	LIST	LIST nl	LIST −nl	LIST nl−	LIST nl1−nl2	REM	'
NEW BRAIN	LIST [−]	LIST nl	LIST −nl	LIST nl−	LIST nl1−nl2	REM	—
ORIC	LIST	LIST nl	—	—	LIST nl1−nl2	REM	Signo equivalente
SHARP MZ-700 (MZ-BASIC)	LIST	LIST nl	LIST −nl	LIST nl−	LIST nl1−nl2	REM	—
SINCLAIR QL	LIST	LIST nl	LIST TO nl	LIST nl TO	LIST nl1 TO nl2	REMark	—
SPECTRAVIDEO	LIST	LIST nl	LIST −nl	LIST nl−	LIST nl−nl2	REM	—
ZX-SPECTRUM	LIST	—	—	LIST nl	—	REM	—

nl = Número de línea. nl1 = Número de línea inicial. nl2 = número de línea final.

FORMULACIONES DE LOS COMANDOS

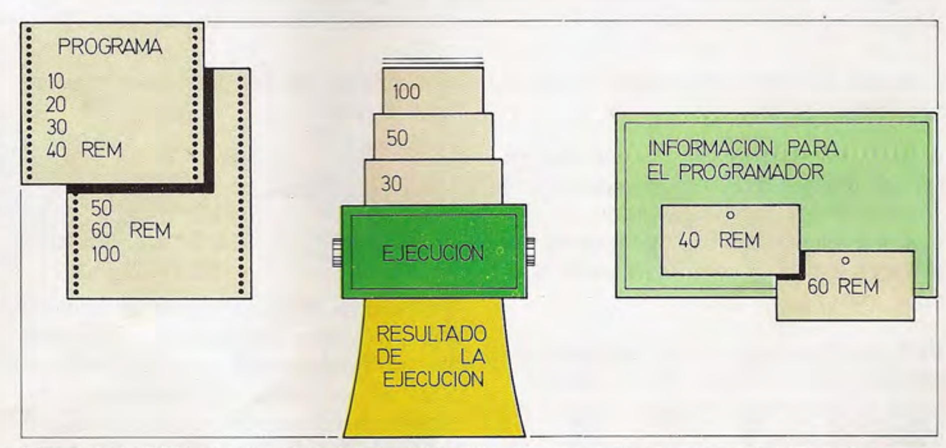
LIST: Lista el programa completo. LIST nl: Muestra en pantalla de línea solicitada. LIST−nl: Presenta las líneas del programa comprendidas desde la primera hasta la especificada. LIST nl−: Lista desde la línea indicada hasta el final del programa. LIST nl1−nl2: Muestra las líneas del programa comprendidas entre las dos especificadas, ambas incluidas. REM: Introduce un comentario en el programa. La presencia de esta instrucción es ignorada por el ordenador durante la ejecución. Signo equivalente: Signo que puede sustituir a la palabra comando REM.

Resulta obvio que para obtener la visualización de los resultados, hay que apoyarse en instrucciones de tipo PRINT. Su argumento es el que servirá para definir la operación a realizar.

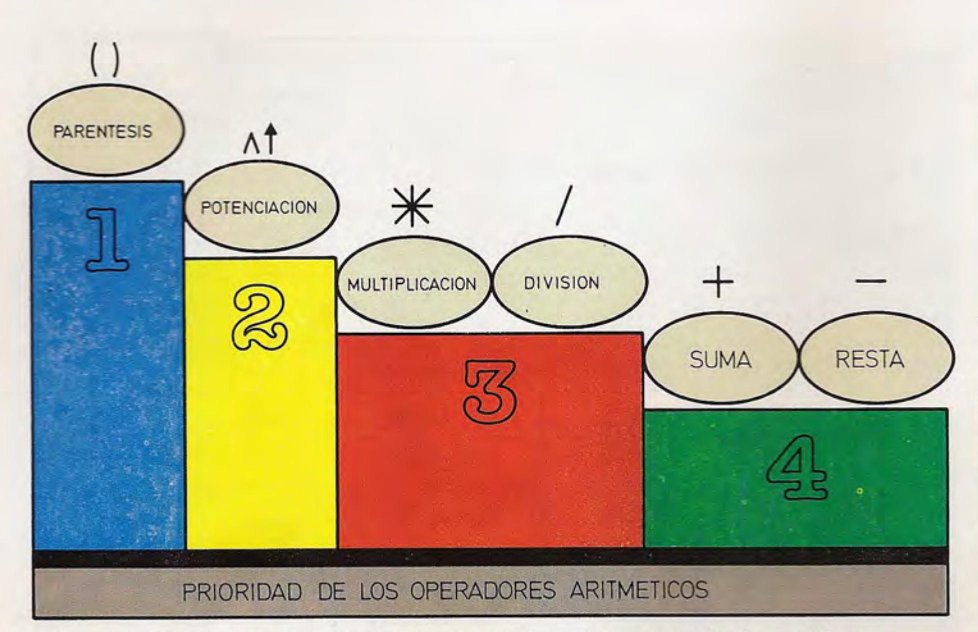
Desde luego, es posible definir operaciones combinadas en las que intervengan varios datos y operadores.

El empleo de varios operadores aritméticos dentro de una misma expresión se rige por las siguientes normas:

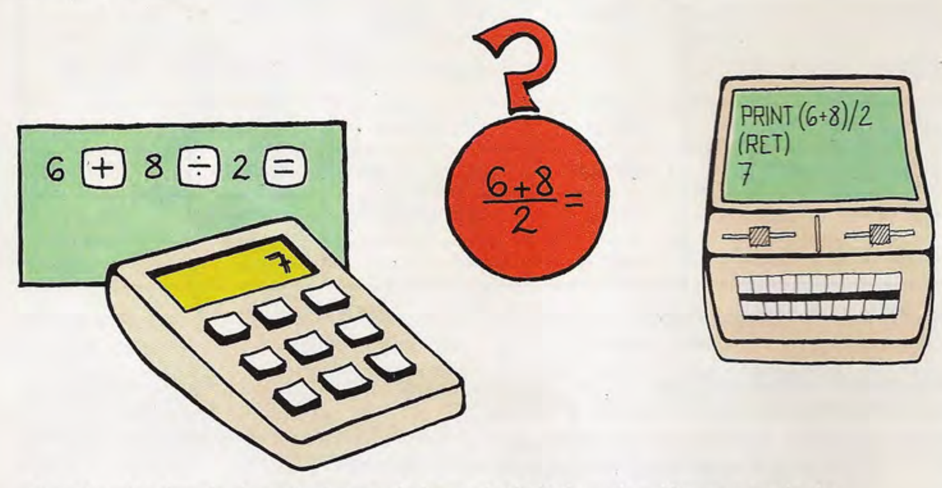
- Dentro de cada expresión las operaciones se ejecutan siguiendo un orden perfectamente establecido: primero se opera la elevación a potencia, luego la multiplica-



El comando REM permite al programador introducir comentarios dentro de los programas. Estos no serán ejecutados por el ordenador.



Al encontrarse con expresiones en las que se combinan datos por medio de operadores aritméticos, el ordenador ejecutará las operaciones de acuerdo a la prioridad establecida por el lenguaje BASIC. Las operaciones encerradas entre paréntesis son las que gozan de máxima prioridad.



El ordenador dotado de un intérprete de lenguaje BASIC puede utilizarse a modo de calculadora. Para ello hay que recurrir al empleo de instrucciones formuladas en modo directo. Por supuesto, en cada caso hay que respetar el modo de introducción adecuado.

ción y/o división y, finalmente, la suma y/o resta.

- Si es necesario alterar la prioridad de las operaciones, por ejemplo, ejecutar una suma antes de una multiplicación, pueden utilizarse paréntesis. Las operaciones encerradas dentro de paréntesis tienen prioridad máxima.

- Si dentro de una misma expresión intervienen varias operaciones de igual prioridad, el intérprete BASIC las operará de izquierda a derecha.

Los siguientes ejemplos, ilustran la aplicación práctica de las prioridades que impone el BASIC:

```
5+3*2=11          (4+2)↑2=36
4+2/2=5           6+4/2*7=20
4+2↑2=8           6+4/(2*7)=6,29
(5+3)*2=16        9-5*8↑4/2=-10.231
(4+2)/2=3         9-5*8↑(4/2)=-311
```

A la hora de emular el funcionamiento de una calculadora, ejecutando instrucciones en modo directo, puede recurrirse al empleo de variables. Por ejemplo, el siguiente par de instrucciones visualiza el resultado de una suma operada por medio de una asignación:

```
A=2+4 (RT)
PRINT A (RT)
6
■
```

En todo caso, es obvio que las posibilidades de cálculo del BASIC no están concebidas para realizar simples operaciones en modo directo. Su verdadero destino es la entrada en los argumentos de instrucciones indirectas que darán cuerpo a programas adecuados para resolver tareas más completas y evolucionadas.

El programa que sigue constituye un ejemplo, sencillo aunque ilustrativo, de una aplicación de cálculo apoyada en la utilidad de los operadores aritméticos:

```
10 INPUT "PRECIO DEL LITRO DE GASOLINA: ";P
20 INPUT "KILOMETROS A RECORRER: ";K
30 INPUT "CONSUMO POR CADA 100 KMS: ";G
40 GA=P*K*G/100
50 PRINT "EL GASTO EN GASOLINA ES DE: ";GA;"PESETAS"
60 END
```

El cometido del programa consiste en calcular cuál va a ser el gasto en gasolina necesario para recorrer un determinado trayecto en automóvil. Los datos que solicitará el ordenador para realizar el cálculo son: precio del litro de gasolina, número de kilómetros a recorrer y consumo de gasolina por cada 100 Kms. (por supuesto, del automóvil que vaya a utilizarse en el viaje). Estos tres datos, deben introducirse a medida que los solicite el ordenador, según vaya ejecutando las sucesivas instrucciones INPUT (líneas 10, 20 y 30).

```
RUN

PRECIO DEL LITRO DE GASOLINA: ? 80
KILOMETROS A RECORRER: ? 250
CONSUMO POR CADA 100 KMS: ? 10
EL GASTO EN GASOLINA ES DE: 2000
PESETAS
■
```

Logo (3)

TURTLE GRAPHICS: dialogando con la tortuga

La herramienta que brinda el LOGO para crear dibujos es la *tortuga:* un simpático colaborador dispuesto a ejecutar los desplazamientos que le ordene el usuario. En sus evoluciones a través de la pantalla, ésta irá construyendo el dibujo al dejar una huella visible de su trayectoria. En efecto, es como si el inquieto personaje llevara una tiza adosada para perpetuar el rastro de sus movimientos. No cabe duda que la técnica resulta didáctica e incluso divertida, además de útil.

La mayor parte de las órdenes que entiende la tortuga son de tipo *comando.* Si bien, también existen operadores que reflejan las condiciones en las que está actuando la tortuga (posición, color, etc.). Más adelante se analizarán incluso algunas posibilidades avanzadas que amplían la eficacia de esta técnica: operación con varias tortugas simultáneamente, o transformación del aspecto de la tortuga a base de «disfrazarla».

Antes de empezar con el trazado de un dibujo, es ineludible observar el aspecto de la tortuga. Es importante distinguir la orientación de su cabeza. Este miembro es fundamental para el movimiento, puesto que indica la dirección en la que se desplazará el personaje. Una opción útil al respecto es pasar a la modalidad de pantalla partida, introduciendo a través del teclado la orden SS. La tortuga pasará a ocupar el centro de la pantalla. Situación en la que será inmediato comprobar cuál es su orientación.

En el instante inicial, la tortuga debe encontrarse mirando hacia el límite superior de la pantalla. Si su disposición fuera otra, será preciso llevarla al estado inicial tecleando la orden CS.

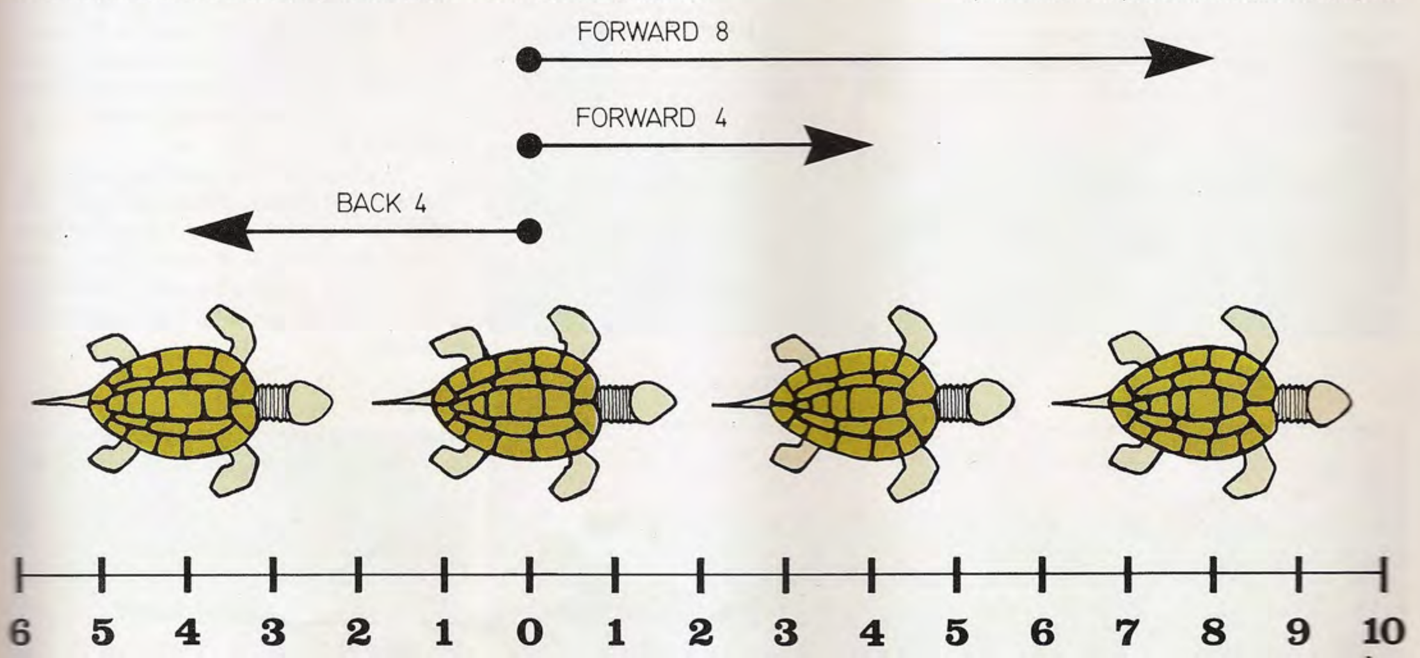
APRENDIENDO A ANDAR

Los desplazamientos más elementales de la tortuga son, naturalmente, los de avanzar o retroceder a lo largo de la dirección marcada por su eje longitudinal. Las órdenes que se ocuparán de instruirla al efecto son las siguientes:

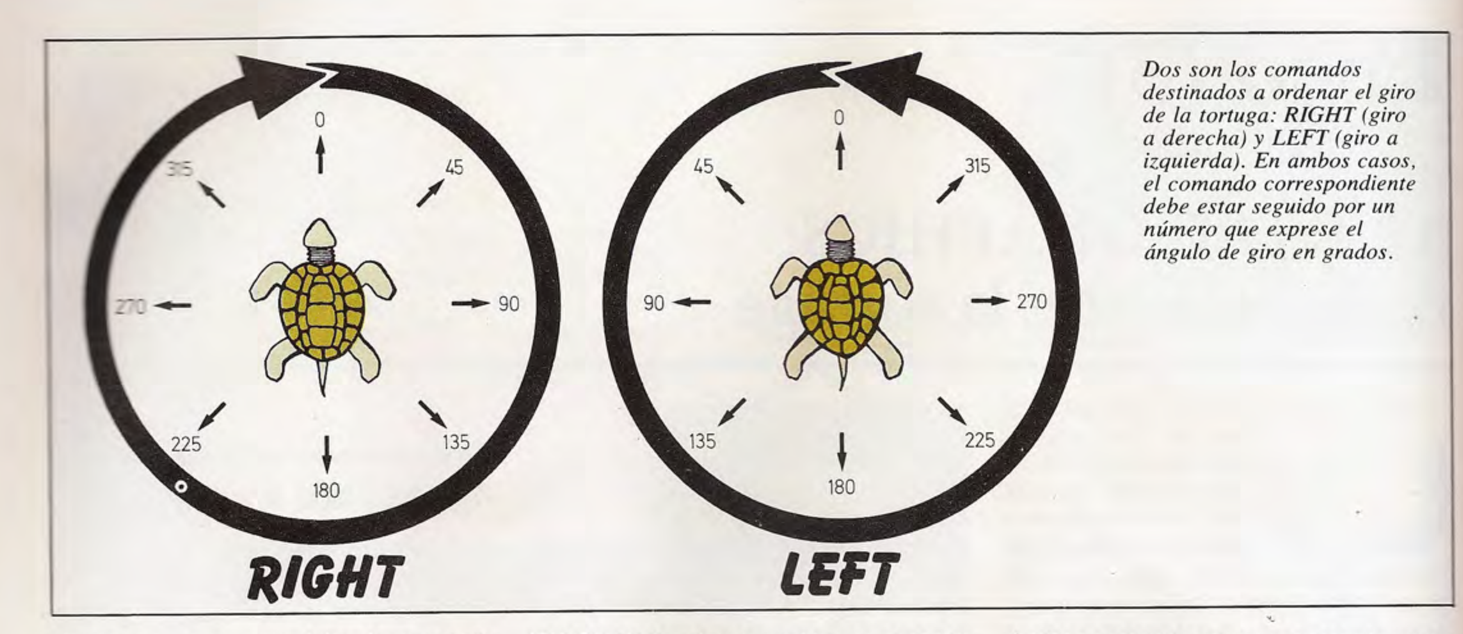
FORWARD: *avance*

BACK: *retroceso*

En ambos casos, ya sea el desplazamiento hacia adelante o hacia atrás, hay que indicar al quelonio el número de posi-



Avance y retroceso (FORWARD Y BACK); estas dos simples acciones son el punto de partida de todo un vocabulario de órdenes que instruirán a la tortuga para la confección de dibujos en la pantalla.



Dos son los comandos destinados a ordenar el giro de la tortuga: RIGHT (giro a derecha) y LEFT (giro a izquierda). En ambos casos, el comando correspondiente debe estar seguido por un número que exprese el ángulo de giro en grados.

ciones o «pasos» que debe dar antes de detenerse. Este número se introducirá tras el comando oportuno.

Por ejemplo, con la orden FORWARD 50, la tortuga se desplaza 50 posiciones hacia adelante, dejando impreso el rastro de su trayectoria. Si queremos que retroceda hasta ocupar de nuevo su posición original bastará con ordenar un desplazamiento hacia atrás del mismo número de posiciones: BACK 50. Si se introduce de nuevo la instrucción BACK 50, la tortuga retrocederá hacia el borde inferior de la pantalla visualizando su huella. Un nuevo FORWARD 50 devolverá a la tortuga al centro de la pantalla, con lo que se habrá dibujado una línea vertical de 100 posiciones o unidades de desplazamiento: 50 hacia arriba y otras cincuenta por debajo de la posición original que ocupaba la tortuga.

Pasemos a movimientos más complejos. ¿Es posible ordenar a la tortuga que cambie la dirección de desplazamiento? En efecto, basta sólo con instruirla para que realice el oportuno giro sobre sí misma. Los comandos al efecto son:

RIGHT: *derecha* y
LEFT: *izquierda*

Ambos comandos deben acompañarse de un dato de entrada que señale los grados del giro (recuerde que un giro de 360 grados equivale a una vuelta completa.

La orden RIGHT 90 dejará a la tortuga mirando hacia la derecha, mientras que un giro ordenado con LEFT 90, devolverá a la tortuga a la orientación de partida (apuntando hacia la zona superior de la pantalla).

Supongamos que a partir de la posición de partida, ordenamos a la tortuga que ejecute la orden RIGHT 90 y tras ésta, la orden FORWARD 50. El resultado será la aparición en la pantalla de una línea horizontal de 50 puntos hacia la derecha, a partir del centro de la pantalla. Si queremos prolongar la línea horizontal por la izquierda, en otras cincuenta posiciones, será suficiente con introducir ahora la orden BACK 100.

¿Cómo devolver ahora a la tortuga a su posición de partida? Nada más fácil; aunque es necesario comunicarle dos órdenes: FORWARD 50 (regresa al punto central) y LEFT 90 (gira noventa grados para apuntar de nuevo al borde superior de la pantalla).

①	②	③	④	⑤
POSICION INICIAL	RIGHT 45	FORWARD 60	BACK 120	FORWARD 60
	45°			

⑥	⑦	⑧	⑨	⑩
LEFT 270	BACK 60	FORWARD 120	BACK 60	LEFT 135
270°				135°



Secuencia de órdenes que instruyen a la tortuga para que dibuje un aspa.

RIGHT 45
FORWARD 60
BACK 120
FORWARD 60
LEFT 270
BACK 60
FORWARD 120
BACK 60
LEFT 135

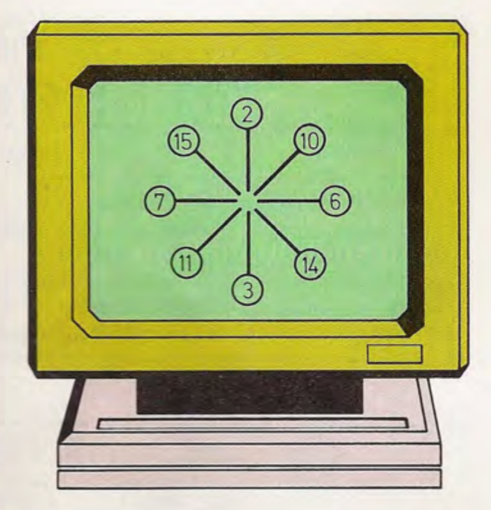
Sin lugar a dudas, el método es muy simple. La figura adjunta muestra la serie de órdenes que educarán a la tortuga para que dibuje un aspa; un ejemplo que resume el empleo de los cuatro comandos presentados.

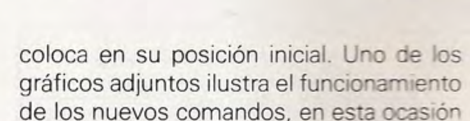
RETORNO AL ORIGEN

Para devolver a la tortuga en la posición inicial, se han utilizado hasta ahora los comandos de movimiento y giro. Este es un método algo engorroso, puesto que las trayectorias de retorno dependen de la posición que ocupe la tortuga, distinta en cada caso. Un método bastante más cómodo para devolver al simpático quelonio a la posición de origen es la que brinda el comando HOME. Su ejecución obliga a la tortuga a regresar al origen, volviendo a la orientación de partida. Por supuesto, la línea que corresponde a la trayectoria de retorno quedará impresa en la pantalla.
Si lo que se desea es volver al principio borrando lo dibujado, hay que optar por otro comando CS (Clear Screen). Este elimina los rastros dejados por la tortuga y la coloca en su posición inicial. Uno de los gráficos adjuntos ilustra el funcionamiento de los nuevos comandos, en esta ocasión dibujando un asterisco.

Un nuevo ejemplo lo aporta el programa adjunto, cuya ejecución hará que la tortuga dibuje un asterisco sobre la pantalla.

```
1  CS
2  FORWARD 50
3  BACK 100
4  HOME
5  RIGHT 90
6  FORWARD 50
7  BACK 100
8  HOME
9  LEFT 45
10 FORWARD 60
11 BACK 120
12 HOME
13 RIGHT 90
14 FORWARD 60
15 BACK 120
16 HOME
```





PINTANDO CON LA TORTUGA

Hasta ahora se han presentado algunos comandos que instruyen a la tortuga para que realice dibujos conexos; o lo que es lo mismo, dibujos continuos en los que todos los trazos están unidos entre sí. Sin embargo, también es posible mover a la tortuga sin dibujar. Para lograrlo, existen algunos comandos que permiten controlar la "tiza" con la que la tortuga pinta sobre la pantalla.
Para desplazar a la tortuga sin que ésta deje el rastro de su trayectoria, es necesario "levantar" la tiza. El comando PENUP es el que ordena tal acción. Una vez ejecutado, los siguientes movimientos de la tortuga no imprimirán trazo alguno en la pantalla.
Cuando haya que pintar de nuevo, habrá que ordenar a la tortuga que "baje la tiza". De ello se ocupa el comando PENDOWN. Este devuelve la tiza a su posición original, dispuesta para trazar la huella del desplazamiento sobre la pantalla. El siguiente ejemplo muestra el empleo de ambos comandos:

```
CS
PENUP
FORWARD 50
RIGHT 90
PENDOWN
FORWARD 50
PENUP
HOME
```

El programa ejemplo dibuja una línea en la zona superior de la pantalla, devolviendo a la tortuga a su posición inicial. Las sucesivas acciones de la tortuga al ejecutar el programa, empiezan tras borrar la pantalla (CS) y "levantar" la tiza (PENUP). Acto seguido (FORWARD 50), la tortuga avanzará 50 posiciones en sentido vertical sin imprimir huella. A continuación, girará 90°

Las órdenes HOME y CS devuelven a la tortuga al origen o posición de partida. La diferencia entre ambas radica en que CS borra la pantalla por completo, mientras que HOME no altera los dibujos que pudieran existir en la misma.

TABLA DE ORDENES DEL "TURTLE GRAPHICS" (1)		
Instrucción	Cometido	Operador/comando
FORWARD <número>	Avance de la tortuga	Comando
BACK <número>	Movimiento hacia atrás	Comando
RIGHT <grados>	Giro a la derecha	Comando
LEFT <grados>	Giro a la izquierda	Comando
HOME	Regreso al centro de pantalla	Comando
CS	Borrado de pantalla y retorno al origen	Comando

TABLA DE ORDENES DEL "TURTLE GRAPHICS" (2)		
Instrucción	Cometido	Operador/comando
PENUP	'Levanta" la tiza	Comando
PENDOWN	"Baja" la tiza	Comando
PE	Activación del borrador	Comando
PX	Activación de tiza de color inverso (negativo)	Comando
PEN	Indica el estado de la tiza o modo de dibujo	Operador
CLEAN	Borra la pantalla dejando a la tortuga en la posición que ocupa	Comando

a la derecha (RIGHT 90) para, de inmediato, "bajar" la tiza (PENDOWN) y dibujar una línea horizontal de 50 posiciones al ejecutar la orden FORWARD 50. Para concluir su actividad, levantará de nuevo la tiza (PENUP) y regresará al original (HOME).

Pero existen aún más posibilidades. El comando PE (Pen Erase) activa el borrador que la tortuga lleva consigo. Por donde pase, después de ejecutar la orden PE, irá borrando lo dibujado.

El último de los comandos de esta categoría es PX. Con este comando la tortuga borrará lo que encuentre pintado y pintará allí donde no encuentre trazos. Hay que tener en cuenta que el efecto de PX es el de alterar el color de tiza, de tal forma que siempre sea el opuesto al color de la zona sobre la que se desliza. En consecuencia, pintará trazos en "negativo", que al superponerse sobre las líneas dibujadas con la tiza en su color normal, las borrarán.

He aquí un nuevo ejemplo:

```
PX
FORWARD 50
RIGHT 90
FORWARD 50
HOME
```

Al ejecutarlo, la tortuga recorrerá el mismo camino que en el ejemplo anterior; si bien, al estar activada ahora la función PX, trazará el "negativo" del dibujo previo y, en consecuencia, borrará la línea horizontal dibujada en el caso anterior.

El siguiente ejemplo recurre a PE para borrar parte del dibujo creado en el ejemplo ilustrativo de PX:

```
PE
FORWARD 25
RIGHT 90
FORWARD 25
HOME
```

Concretamente, la zona borrada coincidirá con la primera mitad de la línea vertical dibujada en el ejemplo anterior.

¿CUAL ES EL ESTADO DE LA TIZA?

En cualquier momento, el programador puede perder la pista y no saber en qué modalidad está utilizando la tiza. Para solventar esta duda, se dispone de la función PEN.

PEN es un *operador* cuya respuesta señala el modo en el que está seleccionada la tiza. Los identificadores con los que responde tal operador constan de dos letras cuyo significado es:

PD: Modo PENDOWN
PU: Modo PENUP
PE: Modo Borrador
PX: Modo Inverso o "negativo".

Como quiera que PEN es un operador, habrá que utilizarlo precedido por un comando; por ejemplo: PRINT PEN, que escribirá en pantalla el identificador del modo en curso.

Por último cabe mencionar la existencia del comando CLEAN. Su función es limpiar la pantalla. Al igual que CS, borra todos los trazos anteriores. Pero a diferencia con aquél, no devuelve la tortuga a la posición de origen. En definitiva, CS equivale a la asociación de los comandos CLEAN y HOME.

Combinando adecuadamente los comandos para el control de la «tiza», es posible crear dibujos discontinuos sobre la pantalla y sacar un mayor partido a las posibilidades de la tortuga.

El mundo del CP/M

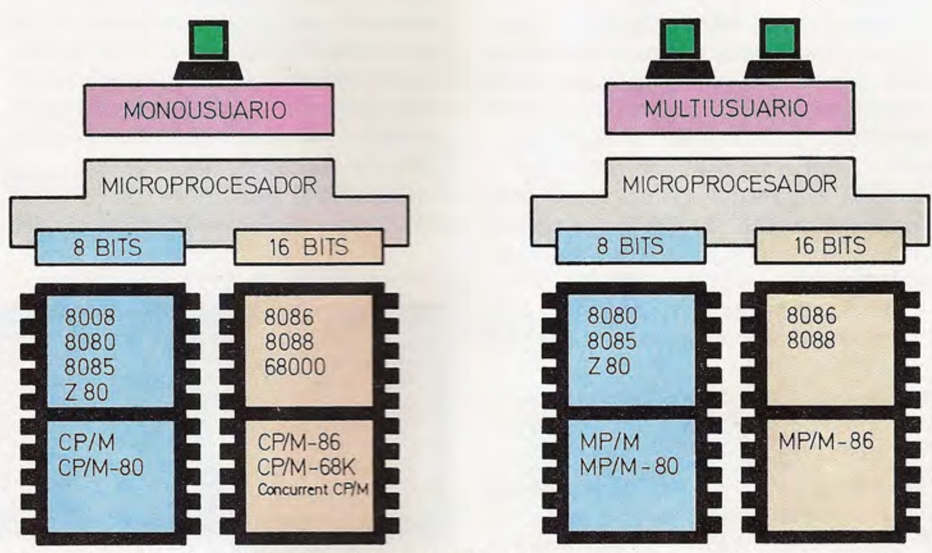
Génesis y características generales de la familia de sistemas operativos CP/M

El acceso a los ordenadores por parte de personas con una formación informática media o básica, se ha hecho posible, en esta última década, gracias al desarrollo de los modernos sistemas operativos. Su presencia en el ordenador permiten al usuario desentenderse de la gestión y control de los dispositivos periféricos que habilitan la comunicación hombre/máquina (pantalla, teclado, impresora...), de los dispositivos destinados al almacenamiento masivo de información (unidades de cinta, disco...), y de los recursos a asignar a las distintas tareas a procesar, así como de la gestión de las prioridades otorgadas a las tareas que aguardan su turno de ejecución.

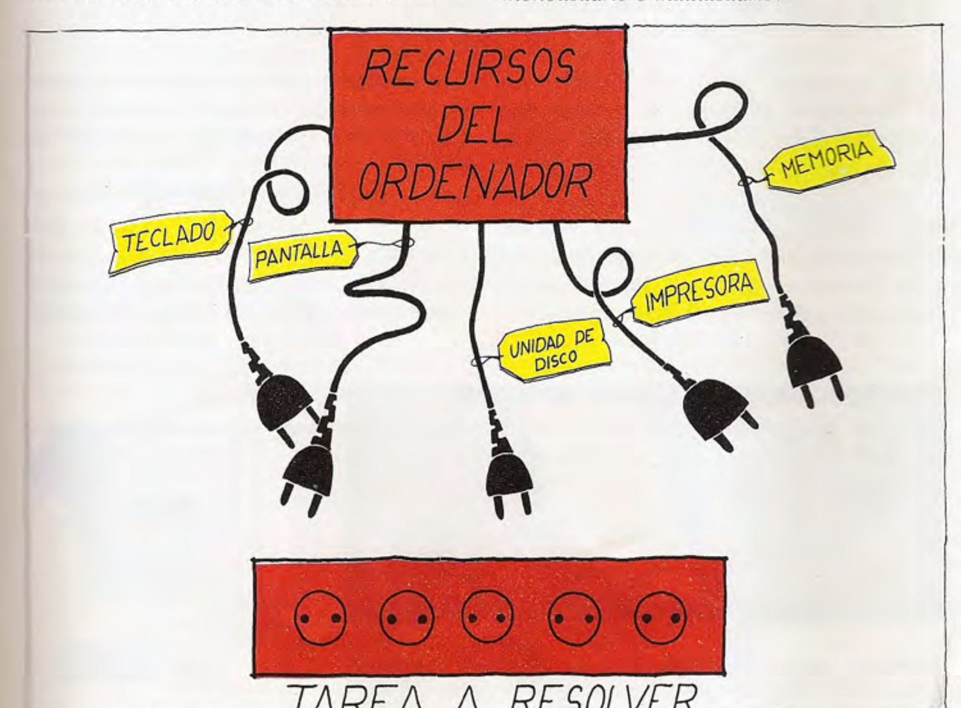
En consecuencia, el usuario puede sustraerse de estas funciones, que aunque son de una importancia capital, pueden llegar a constituir una labor más que tediosa, sobre todo para el usuario que toma su primer contacto con el ordenador.

Gracias a esta inestimable ayuda, el usuario puede colocarse en un nivel de abstracción tal, que le permita realizar una gestión transparente de la máquina. Con ello, podrá dedicar su principal esfuerzo al desarrollo de aplicaciones, sin que sea necesario un aprendizaje largo y exhaustivo de la máquina sobre la que va a trabajar.

El primer sistema operativo, de vocación generalizada, que vio la luz en el campo de los microordenadores fue el CP/M. Actualmente, junto con sus diversas variantes, es uno de los sistemas operativos más difundidos en el mundo de la microinformática.



Dentro de la familia CP/M caben distintas versiones de este sistema operativo: para equipos basados en microprocesadores de 8 ó de 16 bits, y adecuadas para el trabajo en régimen monousuario o multiusuario.





Una de las misiones encomendadas al sistema operativo es el control y la asignación de los recursos de la máquina a las tareas a resolver.

EL NACIMIENTO DEL CP/M

A principios de los años 70, la informática se reducía aún al entorno de las grandes máquinas. Ordenadores con una gran capacidad de cálculo y considerable velocidad, aunque voluminosos y cuyo empleo sólo estaba al alcance de unos pocos elegidos.

A partir de esta fecha empezó a cobrar auge la idea de sacar a la informática de su mundo cerrado y crítico, para acercarla al profesional que necesitara de su potencia de cálculo, pero que no tuviera a su alcance un gran ordenador.

En 1973, Gary Kildall, un empleado de la firma americana Intel, comenzó a diseñar un sistema operativo basado en los siguientes criterios:

- Destinado inicialmente a equipos basados en microprocesadores de 8 bits, del tipo 8008 y 8080.
- Con posibilidad de gestionar el almacenamiento de programas y archivos de datos en memorias auxiliares de bajo coste (en los denominados diskettes, floppy-disks o discos flexibles).
- Que contemplara la creación de un entorno lógico capaz de cubrir la parte física del ordenador (hardware) y que ofreciera al programador toda una colección de funciones que facilitaran la completa comunicación entre el usuario y la máquina. Al mismo tiempo, debía permitir un acceso transparente a los diversos dispositivos periféricos.

Unos años más tarde, en torno a 1976, Gari Kildall creó Digital Research, la firma que, a partir de tal fecha, es la responsable del desarrollo y comercialización del CP/M.

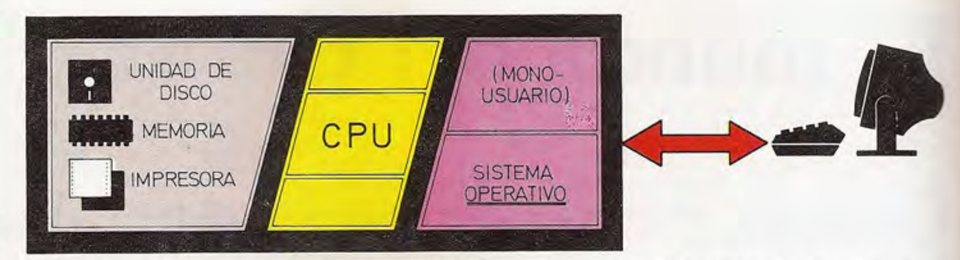
DESARROLLO DEL CP/M

La gran aceptación de este sistema operativo en el mercado de los microordenadores, tiene su constatación en el hecho de que, en nuestros días, se ha convertido en un verdadero estándar. Con un número de usuarios que ha crecido vertiginosamente, bajo el empuje derivado de su aceptación por un gran número de los principales fabricantes de microordenadores, por ejemplo: Digital Equipment, Hewlett Packard, NCR, Texas Instruments, Altos, Bull, Olivetti, Sperry, Intel, ICL, Toshiba, ITT, Xerox, e incluso IBM.

El desarrollo de programas para el sistema operativo CP/M no ha quedado a la zaga. En la actualidad, es ingente la biblioteca de lenguajes, utilidades y software de aplicación destinada al CP/M. En ella se encuentran macro-ensambladores, intérpretes y compiladores de BASIC, PASCAL, COBOL, FORTRAN, LISP y otros lenguajes de alto nivel; editores de líneas y de pantalla y un dilatado abanico de paquetes de aplicación, verticales y horizontales, destinados a casi cualquier ámbito y sector de actividad.

La evolución de este sistema operativo no sólo se ha manifestado en el aspecto cuantitativo con su rápida implantación en el mercado; la gestación y puesta en práctica de la nueva filosofía microinformática, incorporando los conceptos de multiusuario y multitarea, y el nacimiento de los nuevos microprocesadores de 16 bits, ha tenido también un eco inmediato en el CP/M, determinando una importante variación cualitativa.

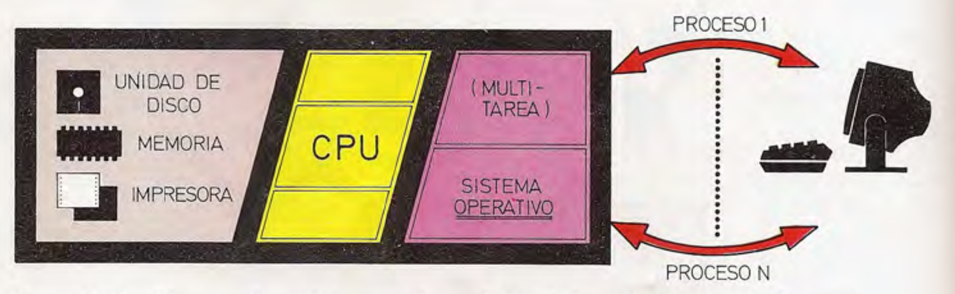
Los conceptos de multitarea y multiusuario parten de la posibilidad de que un solo microprocesador (C.P.U.) atienda a varios usuarios y curse varias tareas. Ello se consigue mediante un mecanismo de asigna-



En régimen "monousuario", sólo es posible la comunicación simultánea de un solo usuario con la máquina; el sistema operativo asignará a éste todos los recursos del ordenador.



La posibilidad de acceso simultáneo al ordenador por parte de varios usuarios se consigue en las configuraciones de tipo "multiusuario". En este caso, el sistema operativo reparte los recursos entre los diversos usuarios que comparten la comunicación con la máquina.



En modo "multiproceso", un único usuario puede beneficiarse de la ejecución de varios programas a la vez: uno de ellos ejecutado de forma interactiva y los restantes en modo "batch" o transparente al usuario.

ción de intervalos de tiempo, en los que la unidad central de proceso dedica su atención a un determinado usuario o tarea específica. Esta técnica denominada "Time slicing", permite tiempos de utilización distribuida de la CPU de unos 20 milisegundos como máximo por usuario o tarea en cada ciclo de atención. De esta forma, el usuario tiene la impresión de que los recursos de la máquina son exclusivamente suyos.
El conjunto de estas innovaciones conceptuales, queda recogido en el sistema operativo MP/M (del inglés "Multi Programming/Monitor"). Una variante, basada en la misma estructura que el sistema operativo CP/M, aunque incorporando las nuevas herramientas que permiten explotar los conceptos de multiproceso y multiusuario.
El segundo punto a tener en cuenta es la influencia que tuvo en el CP/M la aparición de los microprocesadores de 16 bits. La respuesta de Digital Research fue inmediata y se plasmó en el desarrollo de los sistemas operativos CP/M-86 (monousuario y monotarea) y MP/M-86 (multiusuario y multitarea) destinados a los microprocesadores 8086 y 8088 de Intel. Ambas versiones permiten al usuario disponer de un mayor volumen de memoria central de acceso aleatorio (RAM) para el almacenamiento de los programas; liberándolo así de la restricción que supone el límite habitual de los 48 Kbytes de memoria RAM (cabe recordar que las 16 líneas del bus de direcciones de los microprocesadores de 8 bits, sólo permiten el direccionamiento directo de $2^{16}=65.536$ posiciones de memoria, y el sistema operativo suele ocupar alrededor de los 16 Kbytes). La superación de esta barrera permite al usuario trabajar con programas más complejos, con los que acometer problemas que hasta el momento estaban vedados a los microordenadores.

HARDWARE PARA CP/M

El empleo del sistema operativo CP/M exige una configuración hardware mínima para que pueda realizar su cometido. Los elementos requeridos son: el *microprocesador,* que ejecuta las instrucciones en código máquina; la *memoria central,* utilizada para almacenar la zona en curso de tratamiento o de ejecución del sistema operativo o de los programas; *la memoria secundaria* o de masa, para el almacenamiento de datos y programas externamente a la unidad central; la *pantalla* y el

Evolución de los sistemas operativos

El desarrollo de los sistemas operativos ha sido paralelo a la evolución de los propios ordenadores, las máquinas a las que están destinados.

- En los albores de la informática, surgió la PRIMERA GENERACION de sistemas operativos cuya característica básica era el *trabajo secuencial,* desglosado en cuatro fases: perforación de las tarjetas con la información de entrada (tarea encomendada a una máquina denominada perforadora), lectura de los paquetes de tarjetas (lectora), ejecución del programa (por parte de procesadores especializados) y salida de resultados (impresoras).

- La SEGUNDA GENERACION se caracterizó por el *tratamiento por lotes.* También era necesario que concluyera el lote de trabajo en curso antes de pasar al siguiente. No obstante, el calculador descargó una gran parte de su actividad accesoria en máquinas especializadas, lo que permitía su completa dedicación a la parte central del proceso: el cálculo. Una máquina especializada leía las tarjetas de datos y programas, y las grababa en una cinta magnética. Esta era procesada por el calculador central que vertía los resultados en una nueva cinta magnética, de cuya impresión se ocupaba una nueva máquina auxiliar. La velocidad y eficacia se incrementó debido a la entrada en escena de las cintas magnéticas.

- El siguiente eslabón de la cadena evolutiva llegó con la *multiprogramación.* Ahora, los ordenadores y los sistemas operativos de la TERCERA GENERACION, permiten que la memoria central del sistema esté ocupada por varios programas. Una vez ejecutado un programa, el propio ordenador pone a trabajar al periférico de salida adecuado, pasando él a ocuparse del tratamiento del siguiente programa. Una vez concluido este segundo trabajo, o cuando ha terminado de realizarse la salida parcial de resultado del primer programa, el ordenador regresa al tratamiento del primero o prosigue su actividad ocupándose del tercer programa almacenado en la memoria central. En esta tercera generación nacen las técnicas de *multiprogramación* (la máquina almacena varios programas en la memoria central, programas que parecen ejecutarse casi simultáneamente) y *multiusuario* (el ordenador distribuye su atención entre varios usuarios que se comunican con la máquina a través de los respectivos terminales; dada la velocidad de tratamiento de información, parece que el ordenador dedica una atención constante a cada usuario).

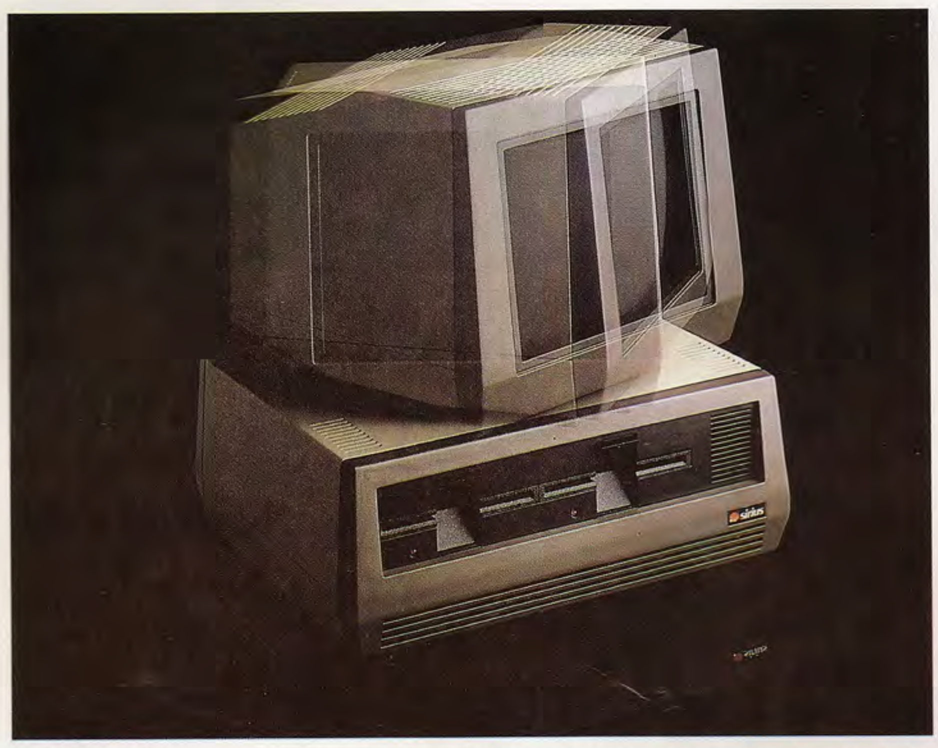
- La CUARTA GENERACION puede definirse como la era de la *informática distribuida y la telemática.* Sintetiza las técnicas propias de la informática, las telecomunicaciones (el teléfono, el satélite...), métodos audiovisuales (TV, video-disco), nuevas técnicas de soportes documentales (microfotografía, facsímil), robótica... Los miniordenadores y microordenadores trasladan la informática a cualquier ámbito de actividad (industrial, profesional, educativo, doméstico...) y nacen las redes para la comunicación entre ordenadores que prestan, en su emplazamiento, una actividad de forma autónoma.


1ª GENERACION TRATAMIENTO SECUENCIAL


2ª GENERACION TRATAMIENTO POR LOTES


3ª GENERACION MULTIPROGRAMACION


4ª GENERACION TELEMATICA



El CP/M es el primer sistema operativo de vocación generalizada que surgió en el campo de la microinformática. En la actualidad cuenta con múltiples versiones adecuadas para muy diversos equipos.

teclado, a través de los que el usuario dialogará con el ordenador, y los *accesos* o "ports" de *entrada/salida,* para la comunicación con los dispositivos periféricos asociados al equipo (impresora, tabla digitalizadora, trazador gráfico...).

Microprocesadores

Los microprocesadores para los que se han desarrollado versiones de este sistema operativo son el 8080, 8085, 8086 y 8088 de Intel y el Z80 de Zilog; todos ellos de 8 bits, excepto el 8086 y 8088 que son de 16 bits. El hecho de que el sistema operativo esté escrito en el lenguaje ensamblador propio de esta familia de microprocesadores, motiva su absoluta dependencia de la máquina. Este punto supone un verdadero óbice que hace que el sistema operativo CP/M sea menos transportable a equipos de distinta naturaleza que otros sistemas operativos, por ejemplo: UCSD y UNIX. Ambos son teóricamente independientes de la máquina, ya que están escritos en un lenguaje de alto nivel: PASCAL para UCSD y "C" en el caso del UNIX.

Memoria central

El tamaño de la memoria central direccionable depende del tipo de microprocesador utilizado. En los microprocesadores de 8 bits el espacio de memoria direccionable llega hasta los 64 kbytes, y hasta más de 1 Megabyte en los microprocesadores de 16 bits. El empleo de estos últimos permite obtener configuraciones con una gran capacidad de memoria libre para el usuario, dado el continuo abaratamiento experimentado por los circuitos integrados de memoria RAM.

Memoria secundaria

Los dos tipos de memorias secundarias o de masa que coexisten, habitualmente, en los equipos con CP/M son las unidades de discos rígidos y las unidades de discos flexibles o "diskettes".

El almacenamiento en diskettes está recomendado para equipos que no exijan gran volumen de información almacenada y para los que los tiempos de acceso, en lectura y escritura, no constituyan un punto crítico. La cantidad de información memorizable en un disco flexible depende de condicionantes de grabación (densidad doble o densidad sencilla) y del número de caras del disco utilizadas al efecto (simple cara o doble cara).

Por contra, las unidades de discos rígidos pueden llegar a almacenar de 5 a 40 Megabytes en dispositivos de tecnología Winchester, con unos tiempos de acceso bastante inferiores a los obtenidos con discos flexibles.

Pantalla y teclado

El diálogo interactivo que permite el S.O. debe estar respaldado por un dispositivo periférico de tipo conversacional, a través del que el usuario sea capaz de comunicarse. La configuración típica consiste en un terminal con pantalla de tipo "CRT" (Cathode Ray Tube) y un teclado semejante al de una máquina de escribir.

Accesos de comunicación

Con estas ventanas al exterior, el microordenador es capaz de comunicarse con dispositivos auxiliares, a través de accesos de entrada/salida en formato serie (habitualmente, según la norma RS/232), o paralelo (interface para impresoras de tipo Centronics, BUS IEE-488...).

El acceso a la unidad central de proceso en los sistemas que operan en régimen multiusuario o multitarea es gestionado por el sistema operativo. Este controla los intervalos de tiempo en los que cada proceso o usuario puede disponer de la atención de la CPU, de acuerdo a las prioridades asignadas.

Software de gestión

Las herramientas de gestión y productividad

Para determinar la zona ocupada por las herramientas de gestión y productividad, dentro del conjunto del software de aplicación, es preciso revisar las distintas clasificaciones que se imponen en este campo.

En primera instancia, y atendiendo a su utilidad específica o generalizada en un determinado marco de tareas, cabe distinguir entre paquetes verticales y horizontales. Otra clasificación, en base a su procedencia, diferencia entre software de creación propia, adquirido como paquete estándar, o encargado para su confección "a medida".

Por último, y adoptando esta vez como criterio el ámbito al que se destina el programa o paquete de programas, se llega a la distinción entre software de juegos o entretenimiento, educación, gestión y productividad, científico/técnico y contabilidad/administración.

El emplazamiento de las herramientas de gestión y productividad hay que precisarlo, en definitiva, dentro de los paquetes comerciales, estandarizados, y de tipo horizontal. Los cinco grupos básicos integrados en esta categoría van a constituir el objeto del presente capítulo.

TRATAMIENTOS DE TEXTOS

La ventaja que supone el empleo de una calculadora, en lugar del lápiz y el papel, a la hora de realizar cálculos matemáticos, es semejante a la contrapartida del ordenador equipado con un tratamiento de textos respecto a la tradicional máquina de escribir.

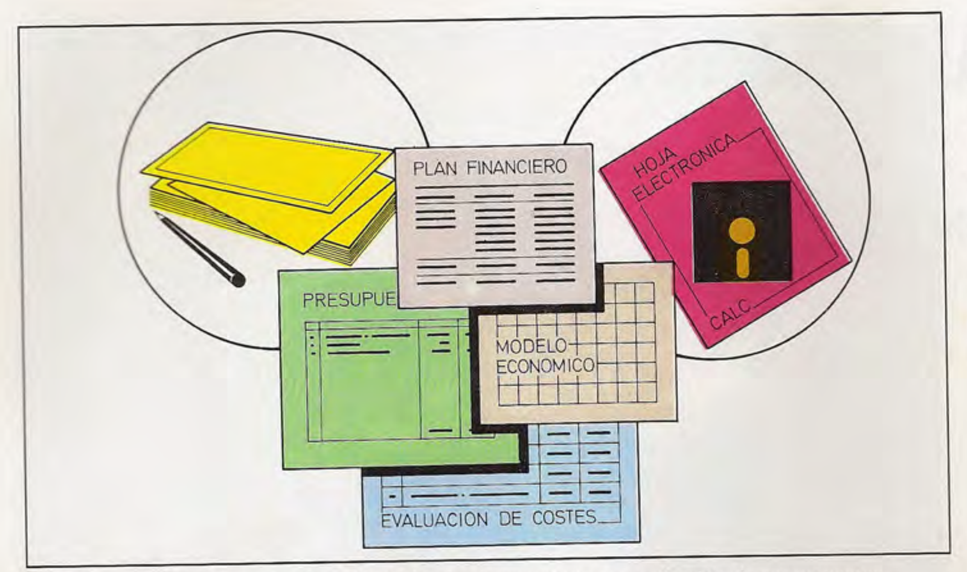
Un paquete para el tratamiento de textos permite todas las funciones propias de una máquina de escribir, si bien, aporta todo un amplio surtido de posibilidades que facilitan, perfeccionan y aceleran la confección de textos. Permite componer el texto en la pantalla, corregirlo con total comodidad, borrar o insertar nuevas palabras y párrafos en cualquier punto del texto editado, redistribuir los bloques del texto y, por supuesto, imprimirlo. En esta última función es, tal vez, donde se encuentra la mayor espectacularidad de los procesadores de textos, puesto que permiten definir tanto el formato de presentación (texto por página, distribución de columnas y márgenes, espaciado entre líneas y párrafos), como, en muchos casos, ➤



El tratamiento de textos por medio de un ordenador sustituye con ventaja a la tradicional máquina de escribir. Cualquier programa actual para el proceso de textos brinda multitud de opciones que apoyan y automatizan la edición de texto escrito.



La ventaja que supone el empleo de una calculadora en lugar del lápiz y el papel a la hora de realizar cálculos matemáticos, es semejante a la contrapartida del ordenador, equipado con un tratamiento de textos, respecto a la tradicional máquina de escribir.



Planificación financiera, evaluación de hipótesis económicas, confección de presupuestos, determinación de costes..., todas éstas son actividades que puede resolver el usuario, rápida y eficazmente, con la ayuda de un ordenador equipado con una "hoja electrónica".

los tipos de letras en los que debe aparecer cada zona del texto (carateres normales, en negrita, cursiva...).

Todo este cúmulo de posibilidades, se completan con la no menos atractiva de almacenar los textos editados para su posterior utilización, modificación o inserción de zonas ya editadas en posteriores documentos.

La utilidad de los paquetes de tratamiento de textos se manifiesta en cualquier situación en la que sea preciso editar o tratar un texto. Ya sea para confeccionar cartas, preparar informes o crear documentos de cuidada presentación.

Existen tratamientos de textos estandarizados, creados para su incorporación a ordenadores personales equipados con un determinado sistema operativo. Entre los más importantes se encuentran: WORDSTAR, EASYWRITER, VISIWORD, APPLE-WRITER, MAGIC WAND, EPISTOLE, WORD PERFECT, MICROSOFT WORD, WORDS, WORDMASTER, TEXTOR... Muchos de ellos disponen de versiones adaptadas al castellano. Los más importantes serán objeto de un detallado estudio práctico dentro de la obra.

TRATAMIENTOS DE TEXTOS		
Nombre	Sistemas operativos	Origen
APPLEWRITER	Apple-DOS, PRODOS, SOS	Apple Computers
EASYWRITER	Apple-DOS, CP/M, MS/DOS	Information Unlimited Software
MAC WRITE	Apple Macintosh	Apple Computers
MAGIC WAND	OASIS	Small Business Applications
MAGIC WINDOW	Apple-DOS, ProDOS, SOS	ARTSCI
TEXTOR	MS/DOS	Talor
VISIWORD	Apple-DOS, CP/M, MS/DOS	VisiCorp
WORD	MS/DOS	Microsoft
WORDSTAR	CP/M, CP/M-86, MS/DOS	Micropro International

HOJAS ELECTRONICAS		
Nombre	Sistemas operativos	Origen
CALCSTAR	CP/M, CP/M-86, MS/DOS	Micropro International
MAGICALC	Apple DOS, ProDOS	ARTSCI
MICROPLAN	CP/M, MS/DOS	Chang Labs.
MULTIPLAN	CP/M, MS/DOS, Apple-DOS y otros	Microsoft
PERFECT CALC	CP/M, MS/DOS	Perfect Software
PLANNER CALC	CP/M, MS/DOS	Target Software
SUPERCALC	CP/M, MS/DOS	Sorcim
T/MAKER II	CP/M, MS/DOS	T/MAKER Corp.
VISICALC	MS/DOS, CP/M, Apple-DOS y otros	Visi Corp

HOJAS ELECTRONICAS

El concepto de hoja electrónica es una de las ideas más revolucionarias que han visto la luz en el terreno del software de aplicación. Hoy en día, su presencia en los ordenadores personales es sustantiva, hasta el punto de convertirse en una herramienta casi imprescindible para ejecutivos, economistas y, en general, para todo aquel profesional cuya actividad exige el uso constante de la calculadora y el papel.

MULTIPLAN, VISICALC, SUPERCALC, CALCSTAR, MAGICALC, T/MAKER II, PERFECT CALC..., son los nombres que identifican a algunas de las hojas electrónicas más populares; nombres que se han integrado en el vocabulario cotidiano de muchos profesionales.

La creación y análisis de modelos financieros; la simulación de hipótesis acerca del funcionamiento de una empresa; el cálculo del precio de un producto, en función de los costes de producción y comercialización; la evaluación de inventarios... Todas ellas son actividades que suelen resolverse a base de proyectar un gran volumen de cálculos sobre una tabla, plagada de casillas, dibujada sobre el papel. Cada vez que se altera el valor de una de las casillas de la tabla, es preciso recalcular el contenido de las restantes casillas afectadas, esgrimiendo la calculadora.

Un ordenador, dotado de un paquete de este tipo, es capaz de generar las tablas adecuadas sobre la pantalla. La intersección de las filas y columnas de la tabla determina las casillas o células, cuyo contenido estará relacionado entre sí por fórmulas o expresiones que definirá el propio usuario. El cálculo del conjunto de casillas será ahora una misión encomendada al ordenador, que realizará automáticamente a partir de los datos proporcionados por el

usuario. Este puede simular cómodamente cualquier hipótesis, sin más que modificar el dato o datos que estime oportuno. El programa se ocupará del resto, recalculando la totalidad de la hoja electrónica para reflejar la nueva situación.

GESTION DE ARCHIVOS Y BASES DE DATOS

El ejercicio de cualquier actividad, profesional o dentro del marco de una empresa, exige el tratamiento de un cierto volumen de información de muy diverso tipo: ficheros de personal, clientes o proveedores; archivo de documentos, correspondencia, informes de productos... La organización y tratamiento de todo este repertorio de datos, por medios informáticos, corre a cargo de los paquetes de esta categoría: los denominados DBMS *(Data Base Management Systems* o sistemas para la gestión de bases de datos).

El software destinado a esa actividad, creado para ordenadores personales, constituye uno de los apartados de mayor resonancia y popularidad. Los paquetes estandarizados orientados a este cometido (dBASE II, dBASE III, PFS File/Report, Omnis, DB Master, Visifile, Friday, Datastar...) ofrecen al usuario todo un conjunto de utilidades destinadas a facilitar la creación de los archivos que integrarán la base de datos y su posterior tratamiento.

La mayor parte de los programas encuadrados en este apartado, permiten al usuario relacionar el contenido de los distintos archivos, realizar operaciones matemáticas con su contenido, clasificar los datos de forma automática y de acuerdo a los criterios que se definan en cada caso, realizar búsquedas selectivas de información y, por supuesto, obtener informes impresos del contenido de los archivos, de acuerdo al formato que establezca el propio usuario.

PAQUETES GRAFICOS

Junto con las hojas electrónicas, el software gráfico constituye el grupo de paquetes de aplicación que goza de las preferencias de los ejecutivos y profesionales en el área de gestión. El software gráfico da entrada a la imagen en las actividades de gestión, sustituyendo los tediosos informes económicos, repletos de cifras, por gráficos de muy diverso tipo: curvas, histogramas, círculos de fraccionamiento proporcional...

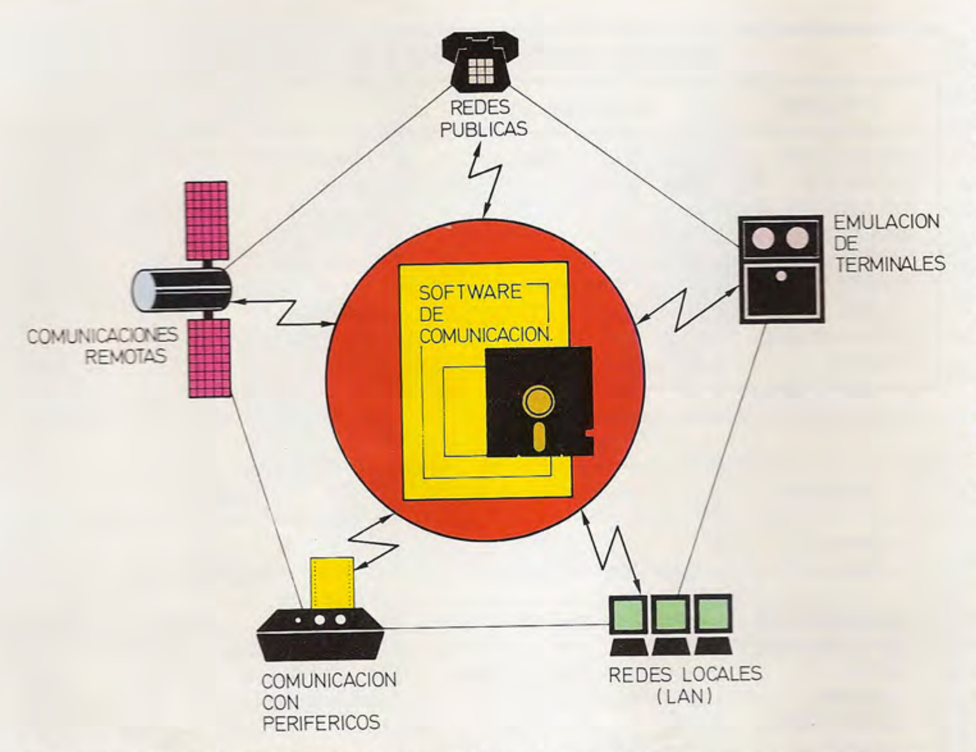
La utilidad de los gráficos de gestión permite el análisis visual de los datos y resultados, mostrando incluso las tendencias

GESTION DE ARCHIVOS Y BASES DE DATOS		
Nombre	Sistemas operativos	Origen
CX BASE 200	Apple-DOS, ProDOS	Controle X
DATASTAR	CP/M, MS/DOS	Micropro
DB MASTER	Apple-DOS, ProDOS	Stoneware
dBASE II/III	CP/M, CP/M-86, MS/DOS	Ashton-Tate
FRIDAY	CP/M, MS/DOS	Ashton-Tate
INFOSTAR	CP/M, CP/M-86, MS/DOS	Micropro
OMNIS	MS/DOS, SOS, UCSD-pSystem	Blyth
PFS File/Report	Apple-DOS, ProDOS, MS/DOS	Software Publishing Corp.
VISIFILE	Apple-DOS, CP/M, MS/DOS	VisiCorp.

GRAFICOS DE GESTION		
Nombre	Sistemas operativos	Origen
BUSINESS GRAPHICS	Apple-DOS, ProDOS	Business and Professional Software
CHARTMASTER	Apple-DOS, ProDOS	Stoneware
CP/M GRAPHICS	CP/M	Digital Research
DATAPLOT	Apple-DOS, ProDOS, SOS	Muse Software
dGRAPH	CP/M, MS/DOS	Fox & Geller
MICROSOFT CHART	CP/M, MS/DOS, Apple-DOS, APPLE Macintosh y otros	Microsoft
PFS GRAPH	Apple-DOS, ProDOS, MS/DOS	Software Publishing Corp.
VISIPLOT	Apple-DOS, CP/M, MS/DOS	VisiCorp
VISITREND	Apple-DOS, CP/M, MS/DOS	VisiCorp



Entre las herramientas de gestión más ampliamente utilizadas cabe señalar a los paquetes para la gestión de archivos y bases de datos. Su utilidad es sustantiva a la hora de automatizar el tratamiento del gran volumen de información que exige el ejercicio de cualquier actividad.



Los límites del ordenador rompen su entorno físico a través de los programas y paquetes para comunicaciones.

que pueden derivar de cada situación analizada. Habitualmente, los paquetes de este tipo permiten al usuario elegir el modo de presentación gráfica entre un amplio abanico de alternativas. La actuación de un paquete gráfico suele estar asociada a una hoja electrónica. De esta forma, los resultados de la hoja son ofrecidos al usuario a través de imágenes que proporcionan una visión casi instantánea



Dentro del software de comunicación caben desde programas adecuados para la comunicación del ordenador con determinados dispositivos periféricos, hasta programas para la creación de redes locales de ordenadores, o paquetes para la comunicación remota, por ejemplo, a través de línea telefónica.

del acontecimiento simulado o del análisis acometido.

La mayor parte de los programas gráficos incorporan la posibilidad de volcar el gráfico en una impresora o en un *plotter* (trazador gráfico), obteniendo, así, copias impresas para acompañar a los informes o ilustrar la documentación relativa al tema.

Algunos paquetes para la generación de presentaciones gráficas son: PFS Graph, VISIPLOT, VISITREND, dGRAPH, CHART MASTER, MICROSOFT CHART, DATAPLOT, BUSINESS GRAPHICS...

SOFTWARE DE COMUNICACIONES

La evolución de los ordenadores personales avanza no sólo en el terreno del tratamiento autónomo de la información, sino también en el ámbito de su comunicación con el mundo exterior.

En este apartado caben diversos tipos de programas y paquetes, destinados a facilitar la comunicación con equipos periféricos, o a establecer el diálogo con otros ordenadores, ya sean del mismo modelo o de distinta categoría y potencia.

De acuerdo a las necesidades de comunicación, los principales grupos de paquetes de comunicaciones son los siguientes:

— Convertidores de protocolo para comunicación con dispositivos periféricos,

— Emuladores de terminales, que permitirán el diálogo del ordenador personal con otros ordenadores de mayor potencia.

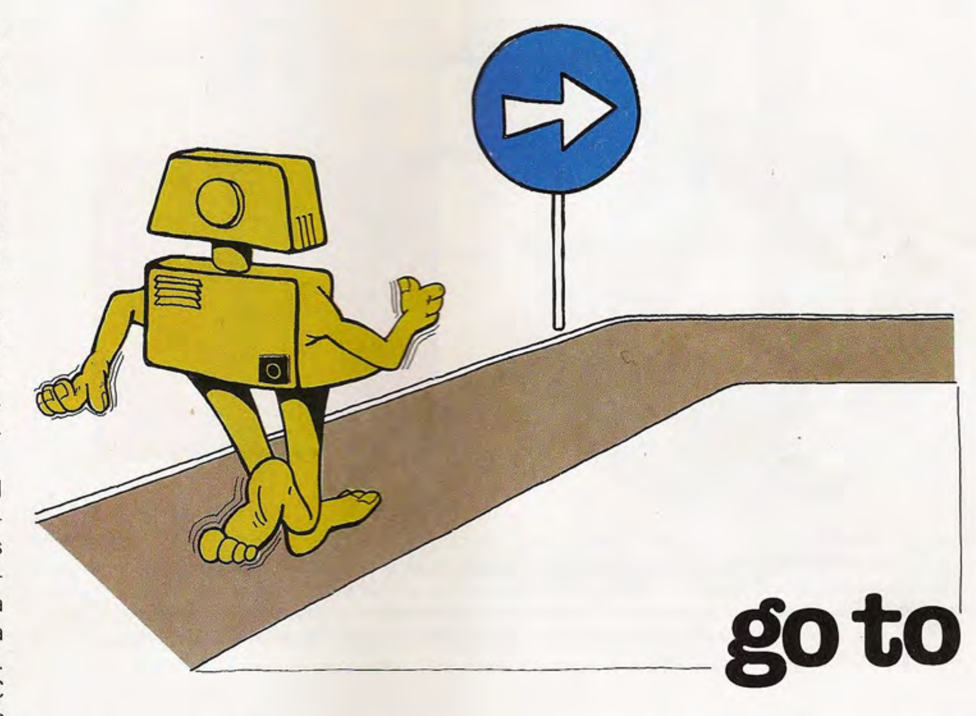
— Paquetes para la creación de redes locales (LAN: Local Area Networks), apoyados por el hardware adecuado. Dentro de un área geográfica limitada, permiten integrar dentro de una red de comunicación a un determinado número de ordenadores del mismo tipo o compatibles. Estos pueden compartir la información puesta en juego por los diversos equipos asociados a la red.

— Paquetes para comunicación remota, sin limitación geográfica; por ejemplo, vía modem, a través de línea telefónica.

Otros paquetes que ya empiezan a entrar en el mercado son los que permiten al ordenador personal acceder a otras fuentes de datos: canales de comunicación específicos, públicos o privados, como, por ejemplo, el sistema "videotext".

Toma de decisiones

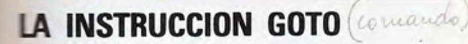
Rupturas de secuencia incondicionales y condicionales



Un programa es, en esencia, un conjunto de instrucciones que detallan un trabajo a realizar por el ordenador. La perfecta conclusión del trabajo programado se obtiene al ejecutar ordenadamente las diversas instrucciones; esto es: en primer lugar, la máquina ejecutará la primera línea del programa, a continuación la segunda y así sucesivamente hasta terminar.

Sabemos ya que el orden de las líneas de instrucción dentro de un programa, viene dado por el *número de línea* que las acompaña. En consecuencia, una línea precederá a otra si su número de línea es inferior.

En algunos casos, es conveniente que el programa no se ejecute de forma totalmente secuencial. Hay ocasiones en las que interesa que tras una cierta instrucción no se ejecute la que está precedida por el siguiente número de línea, sino otra localizada en otro punto del programa. Para facilitar esta posibilidad, el BASIC ofrece medios adecuados para la *ruptura de secuencia.*



LA INSTRUCCION GOTO

La ruptura de la secuencia de ejecución de un programa puede ser, desde luego, obligatoria (incondicional); no obstante, también es posible que la ruptura no deba realizarse en cualquier caso, sino que dependa del cumplimiento de una condición impuesta. Esta distinción está contemplada en el lenguaje BASIC y de ahí que existan instrucciones distintas que permiten la programación tanto de rupturas *incondicionales* como *condicionales.* GOTO es el comando BASIC que permite construir las instrucciones para la ruptura de secuencia o bifurcación incondicional. Su formato general es:

GOTO <número de línea>.

El número de línea que figura como argumento corresponderá a la instrucción que deseamos se ejecute a continuación; esto es: a la instrucción a la que debe realizarse el salto o bifurcación. Una vez ejecutada ésta, el programa seguirá ejecutándose secuencialmente a partir del referido número de línea.

Muchas versiones del lenguaje BASIC permiten incluir dentro del comando

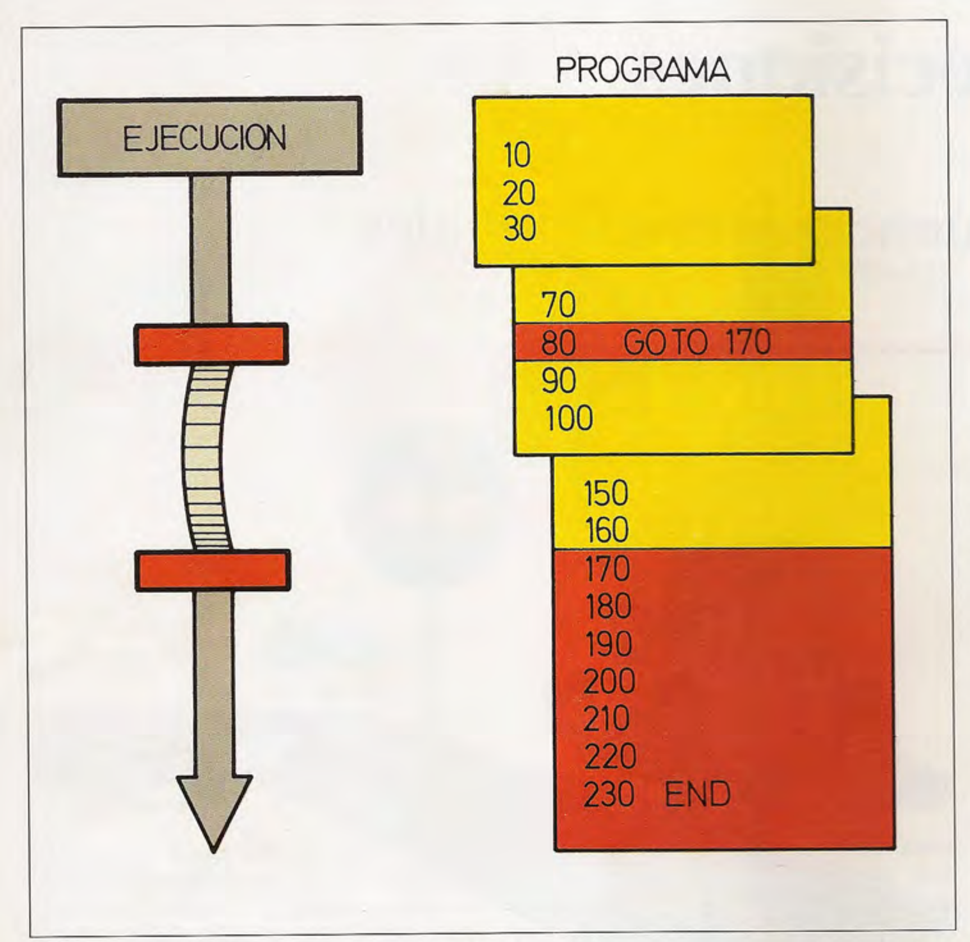
GOTO

Ejecuta un salto al número de línea especificado.

Formato: (Número de línea) GOTO{[<número de línea>] [<expresión>]}

Ejemplos:

```
   GOTO 30
20 GOTO A+30
```



La instrucción GOTO es la herramienta BASIC adecuada para ordenar rupturas de secuencia o bifurcaciones incondicionales. El programa de la figura se ejecutará normalmente hasta llegar a la línea 80. A partir de este punto se rompe la secuencia por efecto de la instrucción GOTO 170; ésta provoca un salto a la línea 170 a partir de la que seguirá ejecutándose el programa.

GOTO una expresión matemática; una vez calculada, su resultado identificará el número de línea al que hay que saltar. Por ejemplo, las dos instrucciones que siguen son equivalentes:

```
GOTO 100
GOTO 50*2
```

Ambas instrucciones ordenan un salto a la línea 100 del programa. La utilidad de la instrucción GOTO se manifiesta especialmente a la hora de programar tareas repetitivas. Por ejemplo, el siguiente programa constituye un bucle sin fin que presentará en la pantalla el texto "CURSO DE BASIC" repetidamente. La ejecución sólo se detendrá cuando el usuario lo ordene por medios directos (accionando la tecla para el abandono del programa en curso, o desconectando el ordenador).

```
10 REM DEMOSTRACION DE GOTO
20 PRINT "CURSO DE BASIC"
30 GOTO 20
■
```

Otro ejemplo, algo más elaborado, recurre a una instrucción GOTO para realizar el cálculo de los cuadrados de los sucesivos números enteros (1, 2, 3...). De nuevo, el programa constituye un bucle sin salida; el usuario debe interrumpir su ejecución por medios directos.

```
10 REM CUADRADOS
20 LET N=1
30 PRINT "EL CUADRADO DE"; N; "ES:"
N*N
40 LET N=N+1
50 GOTO 30
■
```

El programa resulta de lo más simple. La línea 20 da a la variable N el valor inicial uno. Acto seguido está localizada la primera instrucción del bucle repetitivo. Su cometido es mostrar el texto "EL CUADRADO DE", seguido por el valor del número (N) y el texto siguiente ("ES:"); terminando con el cálculo e impresión del cuadrado del número en cuestión (N*N). La línea 40 se encarga de incrementar en una unidad el valor de N; de esta forma, al regresar a la instrucción 30, por efecto del GOTO 30 localizado en la línea siguiente, se procederá al cálculo y presentación en pantalla del cuadrado del siguiente número.

```
RUN
EL CUADRADO DE 1 ES: 1
EL CUADRADO DE 2 ES: 4
EL CUADRADO DE 3 ES: 6
EL CUADRADO DE 4 ES: 16
EL CUADRADO DE 5 ES: 25
EL CUADRADO DE 6 ES: 36
EL CUADRADO DE 7 ES: 49
EL CUADRADO DE 8 ES: 64
```

TOMA DE DECISIONES

Uno de los aspectos que diferencian más notoriamente a los ordenadores del resto de la extensa gama de máquinas electrónicas de cálculo (registradoras, calculadoras de bolsillo, etc...) es, sin duda alguna, la posibilidad de tomar decisiones. A partir de datos suministrados, ya sea por el pro-

grama en curso de ejecución, o bien por el usuario que esté en ese preciso instante frente al teclado de la máquina, el ordenador es capaz de realizar una evaluación de acuerdo a lo consignado en el programa y tomar las decisiones para las que está instruido.

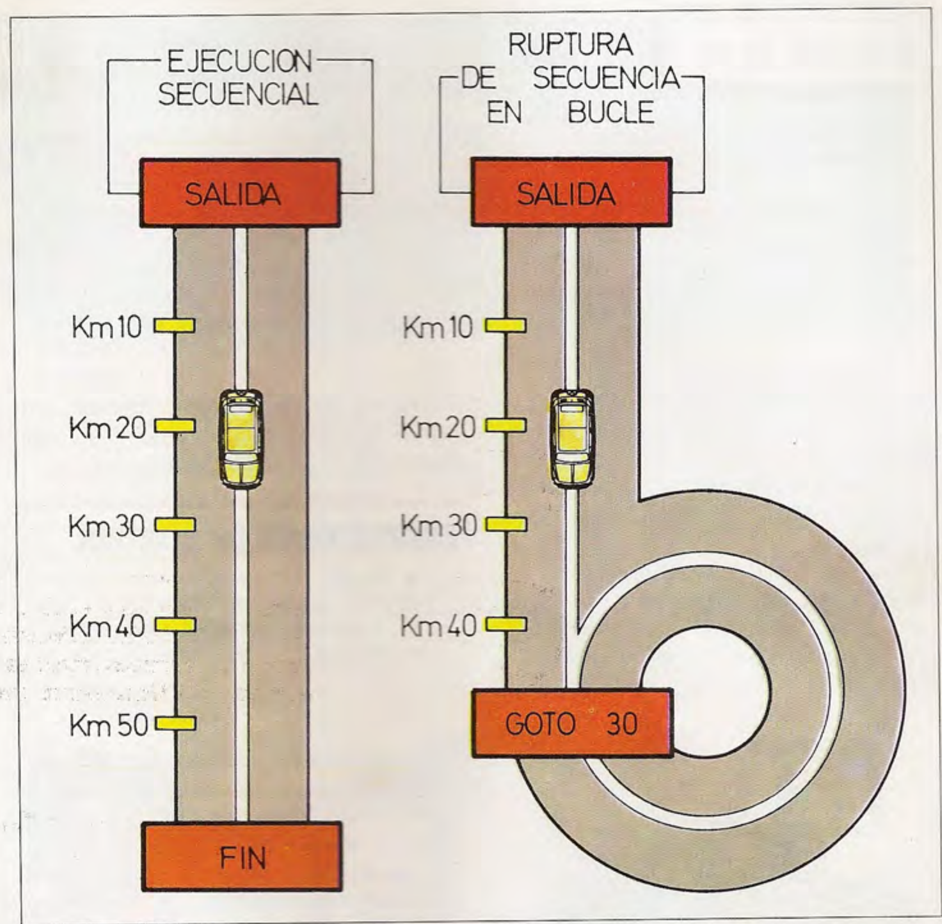
Sin lugar a dudas, éste es un factor que proporciona a la máquina una gran versatilidad, convirtiéndola en un instrumento útil para resolver un ilimitado número de tareas.

Como se recordó al principio del capítulo, un programa en lenguaje BASIC consta de una secuencia de líneas numeradas en orden ascendente. Estas *líneas de instrucción* se irán ejecutando, una tras otra, al ordenar la ejecución del programa. Durante la ejecución pueden presentarse situaciones que exijan procesos distintos. La decisión del camino a seguir puede adoptarla el propio ordenador evaluando una determinada condición; si ésta se verifica puede poner en práctica una determinada acción y, en caso contrario, otra acción distinta. El ordenador debe, por tanto, poseer un mecanismo que le permita en tales situaciones tomar una decisión, de acuerdo con la condición o condiciones establecidas por el usuario.

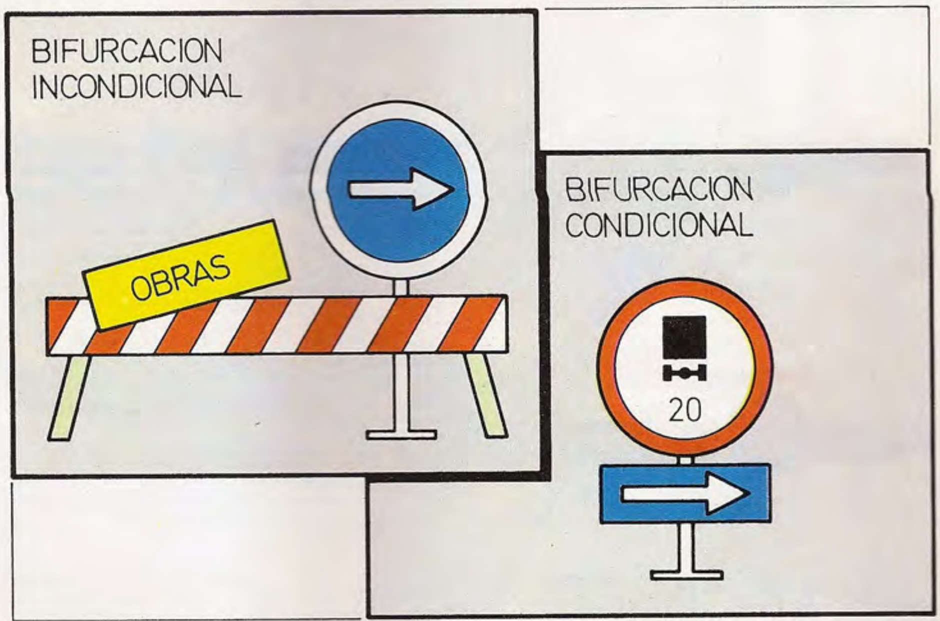
La oportunidad de la toma de decisiones es, pues, indudable. No obstante, hay que partir del hecho de que el ordenador por sí mismo no tiene poder de decisión alguno; debe ser el propio programador quien establezca los términos de la condición a verificar, así como la acción o acciones que debe emprender la máquina como respuesta. Por ejemplo, suponga que se trata de dar al ordenador una serie de cinco números diferentes, para que los sume y presente en la pantalla el resultado. Una primera solución programada podría ser la siguiente:

```
10 LET B=0
20 PRINT "INTRODUZCA UN NUMERO"
30 INPUT A
40 LET B=B+A
50 GOTO 20
■
```

El programa pide la introducción de los sucesivos datos, calcula la suma y la pre-



La posibilidad de romper la secuencia de ejecución de un programa, permite al usuario programar bucles repetitivos.
La presencia de la instrucción de salto hacia atrás (GOTO 30), crea un bucle iterativo que habrá que romper por medios ajenos al programa BASIC.



El lenguaje BASIC ofrece al programador comandos adecuados para ordenar saltos o bifurcaciones incondicionales (GOTO) y condicionales (IF/THEN); estos últimos se apoyan en el cumplimiento o no de una condición impuesta.

IF..THEN..ELSE

Ejecuta las instrucciones que siguen a THEN si se cumple la condición; en caso contrario, se ejecutan las instrucciones que siguen a ELSE.

Formato: (Número de línea) IF <condición> THEN <instrucción 1>[ELSE<instrucción 2>]

Ejemplos:

```
20 IF A>0 THEN P=1
40 IF A>0 THEN P=1 ELSE P=0
50 IF X=5 THEN GOTO 120
```

senta en la pantalla; sin embargo, no está capacitado para detectar cuándo se le han dado los cinco números. La ejecución cierra un bloque continuo, sin detención, excepto por medios ajenos al propio programa.

Es preciso contar con un mecanismo que permita instruir al ordenador para que sea capaz de tomar decisiones. De esta forma, será posible construir un programa que detecte la introducción de los cinco valores y, en ese instante, "decida" presentar la suma de ellos y detener automáticamente el proceso en ejecución.

ESTRUCTURAS DE CONTROL

En la facultad de decisión, no sólo interviene la habilidad del usuario para ingeniar condiciones a evaluar. Tan importante como lo anterior es el tipo de herramientas que brinde el lenguaje BASIC para tal cometido. Herramientas que el programador debe utilizar en la forma apropiada, de acuerdo con las necesidades específicas en cada caso, para programar las tomas de decisión.

Las herramientas de decisión que aporta el lenguaje BASIC se plasman en un conjunto de instrucciones que cabría denominar *estructuras de control*. Las estructuras de control alteran el flujo o la secuencia de ejecución del programa, haciendo que en un determinado momento se ejecuten un conjunto de instrucciones u otro, según se verifique o no una determinada condición.

Existen diversos tipos de estructuras de control, si bien, la principal en el BASIC es la IF/THEN/ELSE.

Su traducción a lenguaje convencional ilustra claramente sus posibilidades: IF (Si) se cumple la condición, THEN (Entonces) ejecutar una determinada instrucción, ELSE (De lo contrario) ejecutar la instrucción siguiente del programa.

Otra estructura de control, muy útil a la hora de tomar decisiones, es WHILE/DO: WHILE (Mientras) se verifique la condición, DO (Hacer) ejecutar la instrucción indicada.

En este caso, la condición que debe verificarse para que se ejecute la acción asociada al DO, se evalúa antes de ejecutar por vez primera la instrucción o acción especificada. Esto significa que si al llegar a una estructura WHILE/DO la condición es falsa, no se ejecutará ninguna vez la instrucción que ocupa la zona DO.

Hay otra estructura de control muy parecida a la anterior aunque con una ligera diferencia. Esta estructura es REPEAT/UNTIL: REPEAT (Repetir) la ejecución de la instrucción indicada UNTIL (Hasta que) se verifique la condición.

El pequeño matiz que la diferencia de la estructura anterior es que, en este caso, primero se ejecuta la instrucción que sigue a la zona REPEAT y, a continuación, se verifica si se cumple o no la condición impuesta. De verificarse, se ejecutará de nuevo la instrucción, y así sucesivamente. En consecuencia, siempre se ejecuta al menos una vez la instrucción asociada a la estructura de control.

Por último, otra estructura de control bastante utilizada en multitud de programas es CASE/OF. Su utilidad se manifiesta a la hora de evaluar una condición y como consecuencia, bifurcar a la instrucción asociada al estado de la condición. En definitiva, permite adoptar una decisión múltiple y ejecutar la acción o respuesta pertinente en cada caso. Las instrucciones a ejecutar se colocan ordenadamente detrás de la zona OF. Al evaluar la expresión asociada a CASE, ha de obtenerse un re-

if/then/else

GRAND PRIX

VUELTAS: 2

RECORD VUELTA: 3' 24"

RECORRIDO TOTAL: 40 V.

REPEAT 

UNTIL 

Estructura de control REPEAT/UNTIL: «repetir la acción indicada (dar vueltas al circuito) hasta que se verifique la condición (fin de las cuarenta vueltas que constituyen el recorrido)».

El microprocesador: un "cerebro" electrónico integrado

La revolución informática es un hecho cuyo origen se encuentra en el vertiginoso avance de la electrónica.

Aún está reciente en el tiempo la época en la que la tecnología electrónica se edificaba por completo sobre las voluminosas y frágiles válvulas de vacío. Pronto, llegaron los semiconductores, con el transistor al frente; su reducido volumen y consumo energético no tardó en relegar a las válvulas casi al olvido.
El tercer peldaño lo aportó el desarrollo de la microelectrónica. Con ella se abría un nuevo horizonte de evolución, tan amplio como creciente ha sido el nivel de integración hacia el que avanzaron los circuitos electrónicos.

Hacia finales de la década de los 60, se crearon los primeros circuitos integrados LSI (Large Scale Integration), de alta escala de integración. Circuitos que alojaban cerca de 10.000 transistores o componentes electrónicos activos en una superficie inferior a un centímetro cuadrado.

El avance no se detuvo y en el mes de noviembre de 1971 irrumpió en los canales comerciales un "chip" cuya referencia era INTEL 4004: el primer microprocesador. En la actualidad —con poco más de una década por medio— se producen chips que integran más de cien mil componentes activos y han visto la luz un considerable número de microprocesadores de 8, 16 e incluso de 32 bits.

En sus orígenes, el microprocesador nació como alternativa a la multiplicidad de circuitos integrados lógicos, orientados a aplicaciones específicas, que debían fabricarse en series reducidas y de dudosa rentabilidad.
Muy pronto, su versatilidad proyectó al microprocesador en el mundo informático. Este pasó a convertirse en la unidad central de proceso de los microordenadores. Un "micro-cerebro" encargado de la manipulación y tratamiento de los datos, del control de las operaciones y de la coordinación de la actividad del conjunto de dispositivos que dan cuerpo al ordenador. Todo ello, de acuerdo a las órdenes que derivan de la ejecución de una secuencia de instrucciones denominada programa.

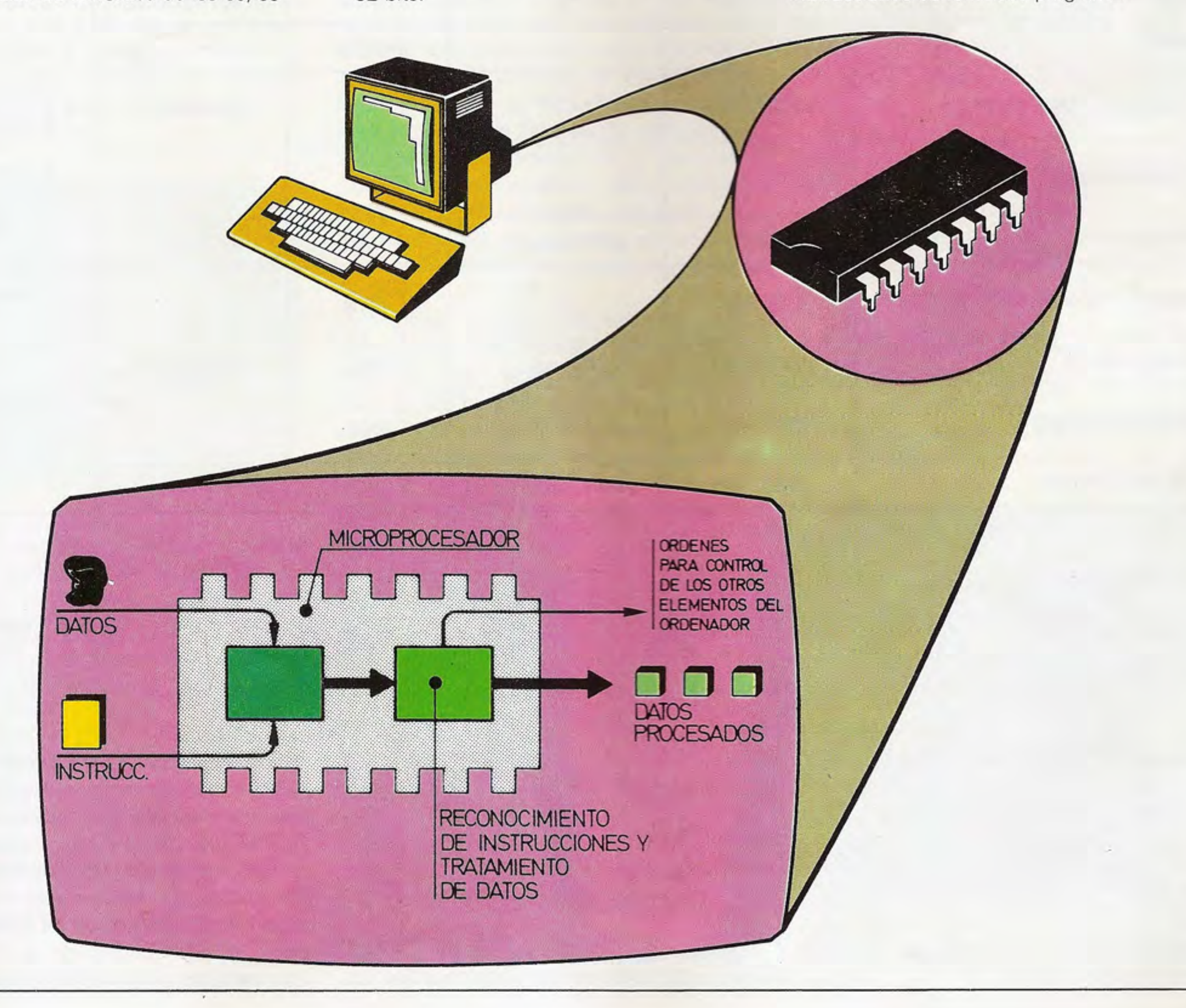


TABLA DE CONVERSION					
ORDENADOR	GOTO		IF/THEN/ELSE		
	GOTO <nl>	GOTO <expr.>	IF/THEN	IF/THEN/ELSE	IF {THEN / GOTO} <nl>
APPLE II (APPLESOFT)	GOTO <nl>	—	IF/THEN	—	IF THEN <nl>
APRICOT (M-BASIC)	GOTO <nl>	—	IF/THEN	IF/THEN/ELSE	IF {THEN / GOTO} <nl>
ATARI	GOTO <nl>	GOTO <expr.>	IF/THEN	—	IF THEN <nl>
CBM 64	GOTO <nl>	—	IF/THEN	—	IF THEN <nl>
DRAGON	GOTO <nl>	—	IF/THEN	IF/THEN/ELSE	IF THEN <nl>
EQUIPOS MSX	GOTO <nl>	—	IF/THEN	IF/THEN/ELSE	IF {THEN / GOTO} <nl>
HP-150	GOTO <nl>	—	IF/THEN	IF/THEN/ELSE	IF {THEN / GOTO} <nl>
IBM PC	GOTO <nl>	—	IF/THEN	IF/THEN/ELSE	IF GOTO <nl>
MPF	GOTO <nl>	—	IF/THEN	—	IF THEN <nl>
NCR DM-V (MS-BASIC)	GOTO <nl>	—	IF/THEN	IF/THEN/ELSE	IF {THEN / GOTO} <nl>
NEW BRAIN	GOTO <nl>	—	IF/THEN	—	IF {THEN / GOTO} <nl>
ORIC	GOTO <nl>	GOTO <expr.>	IF/THEN	IF/THEN/ELSE	—
SHARP MZ-700 (MZ-BASIC)	GOTO <nl>	—	IF/THEN	—	IF {THEN / GOTO} <nl>
SINCLAIR QL	GOTO <nl>	GOTO <expr.>	IF/THEN	IF/THEN/ELSE	—
SPECTRAVIDEO	GOTO <nl>	—	IF/THEN	IF/THEN/ELSE	IF {THEN / GOTO} <nl>
ZX-SPECTRUM	GOTO <nl>	GOTO <expr.>	IF/THEN	—	—

<nl>: Número de línea. <expr.>: Expresión con datos y/o variables.

FORMULACIONES DE LOS COMANDOS

GOTO <nl>: Salto incondicional a la línea cuyo número se indica. GOTO <expr.>: Salto incondicional a la línea cuyo número es el resultado de la expresión. IF/THEN: Bifurcación condicional. Ejecuta la instrucción o instrucciones asociadas a THEN si se cumple la condición expresada en la zona IF. IF/THEN/ELSE: Bifurcación condicional con zona ELSE. La instrucción o instrucciones asociadas a ELSE se ejecutarán en el caso de no verificarse la condición impuesta en IF

IF {THEN / GOTO} <nl>: Posibilidad de omitir una de las palabras clave (THEN o GOTO) cuando la zona THEN contiene una instrucción de salto incondicional.

sultado coincidente con un número entero (1,2,3,4,...). Según sea tal valor, la instrucción OF ejecutada será la que ocupe el primer lugar, el segundo, tercero... Esta estructura de control se tratará posteriormente, en otro punto de la obra, dado que el lenguaje BASIC incorpora una instrucción capaz de generarla directamente: ON/GOTO.

... Y A TOMAR DECISIONES

Aunque hay dialectos del lenguaje BASIC que incorporan en su repertorio los comandos WHILE/DO y REPEAT/UNTIL, lo habitual es que omitan tales comandos y reduzcan su oferta a la instrucción IF/THEN/ELSE. En cualquier caso, las dos estructuras de control mencionadas pueden simularse mediante el empleo adecuado de la instrucción de salto condicional IF/THEN/ELSE. Esta instrucción es una transposición práctica de la respectiva

estructura de control. Por lo demás, y salvo muy pocas excepciones, esta instrucción adopta idéntico formato en casi todos los ordenadores personales presentes en el mercado.

La diferencia más generalizada se encuentra en los dialectos BASIC que omiten la zona ELSE. En todo caso, esta opción queda implícita en la propia estructura IF/THEN: si la condición que se ha de verificar no es cierta, se ejecutará la línea siguiente a la que ocupa la propia instrucción IF/THEN; de ahí que la función ELSE pueda obviarse sin mayores problemas. Incluso aunque esté disponible, el uso de la zona ELSE es opcional, de forma que en la práctica suele omitirse en muchos casos.

Volviendo al ejemplo anterior, ahora el ordenador sí puede estar ya en condiciones para saber cuándo se le han dado los cinco números; es posible programar la toma de decisión adecuada. Así pues, el ordenador podrá evaluar si se han introducido ya los cinco números y, en consecuencia, calcular la suma, visualizar el resultado y detener la ejecución del programa. Veamos cuál será el nuevo aspecto del programa:

```
10 LET B=0
20 LET N=0
30 PRINT "INTRODUZCA UN NUMERO:"
40 INPUT A
50 LET N=N+1
60 LET B=B+A
70 IF N< >5 THEN GOTO 30 ELSE
PRINT "LA SUMA ES=";B
80 END
■
```

En esta ocasión, la variable N lleva la cuenta de cuántos números se van introduciendo en el ordenador; su valor —inicializado a cero— se incrementa en una unidad cada vez que se teclea un nuevo dato. A su vez, la variable B —también inicializada a cero— irá totalizando el valor de la suma de los datos introducidos.

En la línea 70 se comprueba la condición: ¿valor de N distinto de 5? Si la condición se cumple, esto es, si aún no se han introducido los cinco números a sumar, se ejecutará la zona THEN y, en consecuencia, se regresará de nuevo a la ejecución de la línea 30 (GOTO 30). Por el contrario, si la condición deja de cumplirse (N ha alcanzado ya el valor 5), se ejecutará la instrucción asociada a la zona ELSE; ésta presenta en la pantalla la suma total de los cinco datos. Acto seguido, el ordenador pasará a ocuparse de la instrucción 80 que pondrá fin a la ejecución del programa.

El comando IF/THEN/ELSE, está abierto a la construcción de muy diversos tipos de instrucciones de bifurcación condicional. De acuerdo al estado de la condición, puede desencadenar un salto o bifurcación a una determinada línea del programa, o puede dar paso a la ejecución de una instrucción que no implique un salto (INPUT, PRINT, LET...).

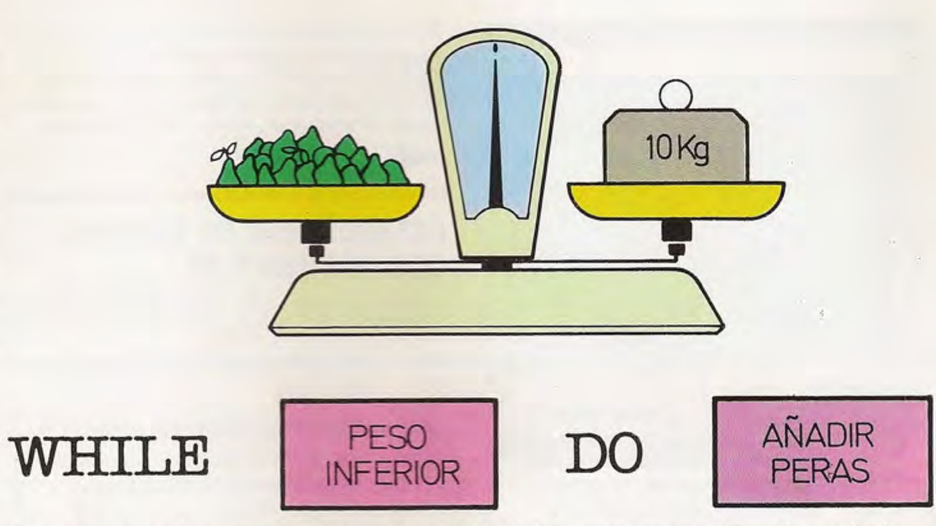
Para dotar de mayor versatilidad a esta instrucción, normalmente, es posible introducir detrás de la palabra THEN, o de la zona ELSE, más de una instrucción, separadas por el símbolo dos puntos (:). Con ello podrá ejecutarse más de una acción como respuesta a una misma decisión adoptada en la zona IF. Algunos intérpretes de BASIC permiten incluso suprimir la palabra THEN cuando ésta se completa con la instrucción GOTO. En tal caso, las dos instrucciones que siguen serán equivalentes:

```
70 IF N< >5 THEN GOTO 30
70 IF N< >5 GOTO 30
```

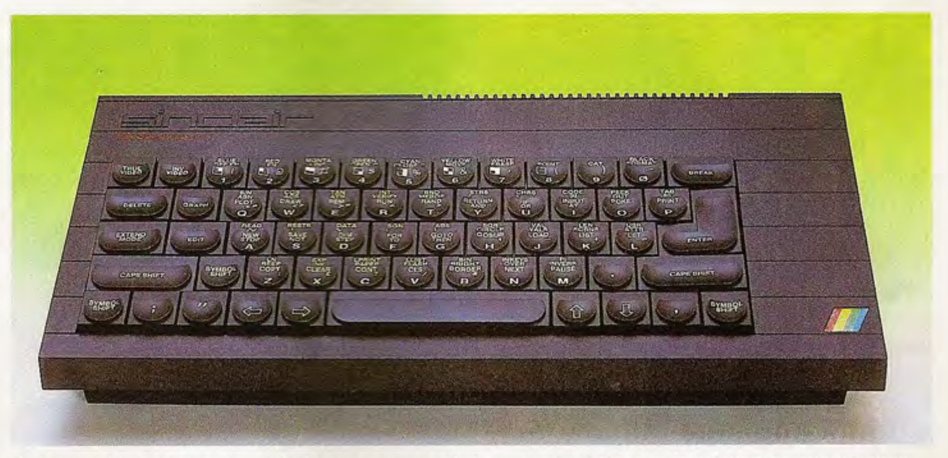
En otros dialectos BASIC, la palabra clave que se puede omitir en tal situación es GOTO. Por ejemplo:

```
70 IF N< >5 THEN GOTO 30
70 IF N< >5 GOTO 30
```

De nuevo, ambas instrucciones serían equivalentes.



Otra estructura de control muy frecuente es WHILE/DO: «mientras se verifique la condición impuesta (peso inferior) ejecutar la acción indicada (añadir peras a la balanza).



La capacidad para tomar decisiones es una de las características más relevantes que otorgan a la máquina los lenguajes de programación. Esta facultad programable distancia al ordenador de otras máquinas electrónicas de aplicación especializada.



Las instrucciones IF/THEN/ELSE (en su formulación completa u omitiendo la zona ELSE) permiten programar tomas de decisión asociadas a bifurcaciones o saltos en la secuencia de ejecución del programa.



ESTRUCTURAS DE CONTROL CON IF/THEN/ELSE

Una vez descrito el funcionamiento de la instrucción IF/THEN/ELSE, es posible llevarla al terreno práctico programando, por ejemplo, las restantes estructuras de control a partir de la propia IF/THEN/ELSE. El ejemplo que sigue simula totalmente el comportamiento de la estructura de decisión WHILE/DO:

La figura ilustra la ejecución del programa que aparece en la pantalla superior. Este incluye los dos tipos de bifurcaciones o saltos: condicional (línea 20) e incondicional (línea 40). Tal como se observa, el programa ejecuta una acción reiterada (acumular sucesivos elementos), hasta que se cumple la condición establecida en la instrucción 20 (N=3); en cuyo caso se produce un salto a la instrucción final.



```
 5 REM TOMA DE DECISION
10 INPUT "¿CUAL ES EL COLOR?"; C$
20 IF C$="VERDE" THEN GOTO 40 ELSE PRINT
   "INTENTELO DE NUEVO"
30 GOTO 10
40 PRINT "¡BRAVO, LO ACERTO!"
50 END
```



Puesta en práctica de una toma de decisión por medio de una instrucción del tipo IF/THEN/ELSE.
El programa formula una pregunta al usuario (¿Cuál es el color?); la corrección de la respuesta se comprueba en la línea 20. Si la respuesta es correcta, la ejecución proseguirá en la línea 40, presentando el mensaje oportuno antes de dar por concluido el programa. Por el contrario, si la respuesta es errónea, la zona ELSE lo comunicará al usuario y la instrucción GOTO 10 de la línea 30 ordenará una nueva ejecución del programa.

```
10 INPUT "INTRODUZCA EL NUMERO:";A
20 IF A=0 THEN GOTO 50
30 PRINT A,2*A,A*A
40 GOTO 10
50 END
```

La actividad del programa consiste, sencillamente, en presentar en la pantalla el número introducido, el doble de su valor y el cuadrado del número en cuestión. Esta tarea se lleva a cabo siempre que el número tecleado sea distinto de cero. Si el número que se introduce es el cero, la condición será cierta y, en consecuencia, el programa concluirá por efecto del GOTO 50 emplazado en la zona THEN.

Cabe observar que si el primer número introducido es el cero, no se ejecutarán ninguna vez las líneas 30 y 40; como se recordará, ésta es una característica inherente a la estructura de control WHILE/DO.

La estructura REPEAT/UNTIL se obtiene fácilmente por medio de una instrucción IF/THEN/ELSE. Una demostración elocuente la brinda el siguiente ejemplo, cuya funcionalidad es análoga a la del programa anterior:

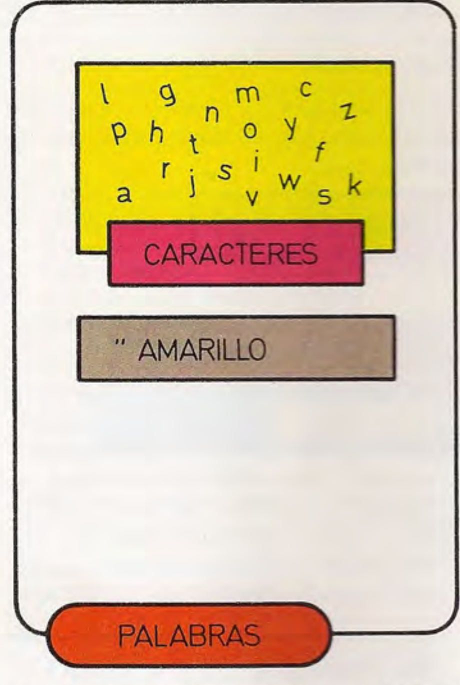
```
10 INPUT "INTRODUZCA EL NUMERO:"; A
20 PRINT A,2*A,A*A
30 IF A< >0 THEN GOTO 10
40 END
```

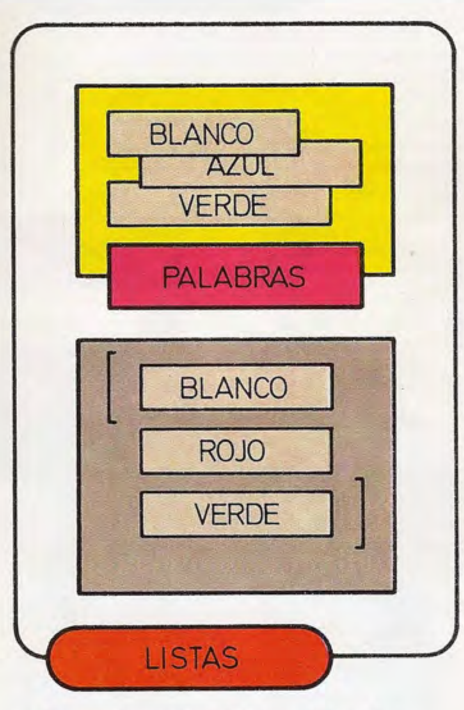
En este caso, aunque el primer número que se introduzca a través del teclado sea el cero, se mostrará en la pantalla el resultado de las operaciones realizadas. Por lo tanto, siempre se ejecutará, al menos una vez, la acción que corresponde a la zona REPEAT; característica ésta peculiar de la estructura de control REPEAT/UNTIL.

Logo (4)

Tratamiento de palabras y listas

Una de las facetas más importantes del LOGO es el tratamiento de palabras y listas. En la terminología de este lenguaje, una palabra es un conjunto de letras, números y/o signos especiales. Dada esta definición, es obvio que cualquier palabra en castellano (o en otro idioma) coincidirá con una palabra LOGO. Cabe recordar que ya en capítulos precedentes se han utilizado palabras para dar nombre a las variables.
A su vez, una lista no es más que un conjunto de palabras. Una frase, un párrafo, o incluso un texto, son ejemplos de listas. El lenguaje LOGO permite manipular con notable soltura ambos tipos de elementos. En el tratamiento intervienen ciertos símbolos separadores que tienen asignada una específica función.





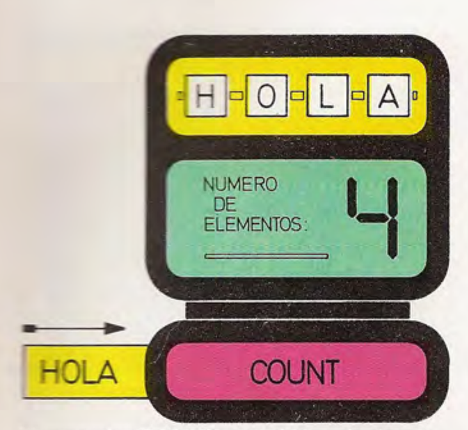
Una de las aptitudes más relevantes del LOGO es el tratamiento de palabras y listas. Las palabras son agrupaciones de caracteres (letras, números y/o símbolos especiales autorizados), mientras que las listas están constituidas por conjuntos de palabras o de otras listas más elementales (sublistas).

SEPARADORES

El separador es un símbolo especial cuya misión es delimitar las distintas partes de una "frase". El principal separador en LOGO es el espacio en blanco. Su presencia revela el fin de una palabra.
Si dos palabras se escriben una tras otra, sin interponer un espacio blanco entre ambas, el intérprete LOGO entenderá que se trata de una sola palabra. Así pues, UNOYDOS es una palabra única, UNO YDOS será interpretado como dos palabras, y UNO Y DOS como tres palabras independientes.
Un ordenador preparado para dialogar en LOGO no será capaz de asimilar, por ejemplo, la orden SUM32. Si en su lugar, se introduce la orden SUM 32, la máquina tomará como primera entrada 32 y será incapaz de encontrar el segundo dato de entrada. La instrucción correcta será, en definitiva, SUM 3 2. Este es un ejemplo que ilustra la importancia que tiene la correcta separación de las distintas zonas que intervienen en una instrucción.

COMILLAS Y CORCHETES

Una palabra puede ser un dato de entrada para una instrucción LOGO. En tal caso, la palabra ha de ir precedida por comillas. Para que el LOGO entienda que UNOYDOS es un dato de entrada, esta palabra debe aparecer como: "UNOYDOS. Por ejemplo, para escribir en la pantalla HOLA, la instrucción adecuada es:
PRINT "HOLA.
Cuando hay que tratar dos o más palabras juntas, éstas deben agruparse en la forma adecuada, formando una *lista:* conjunto de palabras encerradas entre corchetes.
[BUENOS DIAS] es una lista y el ordenador la tratará como si se tratara de una unidad. En cambio, "BUENOS "DIAS son dos palabras independientes: no constituyen una lista y, en consecuencia, el ordenador las procesará por separado. Por ejemplo:
PRINT [BUENOS DIAS]
escribirá en la pantalla la lista completa.



El operador COUNT contabiliza el número de elementos que integran el dato de entrada sometido a examen; éste puede ser una palabra o una lista. En el primer caso se contarán los caracteres que conforman la palabra, mientras que en el segundo serán las palabras y sublistas los elementos contabilizados.

```
PRINT [BUENOS DIAS] (RT)
BUENOS DIAS
```

Una lista puede incluir, a su vez, otras listas de menor entidad o *sublistas*. Por ejemplo:

[HOLA [BUENOS DIAS]]

es una lista de dos elementos: una palabra y una sublista. La sublista en cuestión consta de dos elementos que en este caso coinciden con dos palabras simples. El concepto de elemento queda totalmente clarificado con el uso del *operador* COUNT (contar).

COUNT admite como dato de entrada una palabra o una lista, y devuelve como dato de salida el número de elementos que constituyen la entrada.

Así por ejemplo:

```
PRINT COUNT "HOLA (RT)
4
```

En efecto, el dato de salida o resultado es el número 4, puesto que son cuatro los elementos (caracteres) de la palabra examinada. Si en lugar de aplicar el operador COUNT a una palabra lo hacemos sobre una lista, los elementos a contar no serán ya caracteres, sino palabras y sublistas:

```
PRINT COUNT [BUENOS DIAS] (RT)
2
```

El resultado revela, en efecto, el número de palabras que intervienen en la lista señalada.



FIRST es un operador LOGO que trunca su dato de entrada (palabra o lista) tomando de éste el primer elemento. BUTFIRST realiza el mismo fraccionamiento, si bien, entrega como salida todos los elementos del dato de entrada exceptuado el primero.

Tanto para [ ] (lista vacía) como para " (palabra vacía) el resultado de COUNT es 0.

TRATAMIENTO DE PALABRAS Y LISTAS

El LOGO posee un nutrido grupo de operadores especializados en el tratamiento de palabras y listas. En primer lugar cabe mencionar los cuatro siguientes: FIRST, LAST, BUTFIRST y BUTLAST. Todos ellos son aplicables tanto a palabras como a listas.

FIRST y LAST devuelven, respectivamente, el primer y el último elemento del dato de entrada, mientras que los otros dos operadores hacen exactamente lo contrario; esto es: BUTFIRST elimina el primer elemento y BUTLAST el último. Veamos el siguiente ejemplo:

```
MAKE "LISTA1 [MI NOMBRE ES]
PRINT FIRST :LISTA1
MI

PRINT LAST :LISTA1
ES
```

El programa empieza haciendo uso del comando MAKE para definir la LISTA1. Esta última palabra es el nombre que va a identificar a la variable, mientras que el dato a asignar (la lista de palabras) coincide con MI NOMBRE ES. Acto seguido aparece la instrucción PRINT FIRST :LISTA1, cuya finalidad es mostrar en la pantalla el resultado de aplicar el operador FIRST al contenido de la variable LISTA1 (cabe recordar la importancia de los dos puntos como indicador de variable). Por supuesto, el dato de salida es MI: la primera palabra o elemento de la lista.

La siguiente instrucción ordena la presentación en pantalla del último elemento de la referida lista, para lo cual se utiliza el operador LAST.

Los otros dos operadores presentados aplicados a la misma lista definida, darán como resultado el que muestra la siguiente pantalla:

```
PRINT BUTFIRST :LISTA1
NOMBRE ES

PRINT BUTLAST :LISTA1
MI NOMBRE
```

BUTFIRST presenta el contenido de la lista especificada, una vez suprimido el primer elemento, mientras que BUTLAST muestra la lista en cuestión exceptuando el último de sus elementos.

Las posibilidades de los cuatro operadores descritos no se reducen a la simple actuación independiente. Pueden asociarse para dar un tratamiento más complejo a los datos de entrada.

```
PRINT FIRST FIRST BUTLAST :LISTA1
M
```

Este es un ejemplo algo más complicado que combina la presencia de dos operadores FIRST y uno de tipo BUTLAST. Cabe

recordar que su aplicación es extensiva a listas y a palabras, actuando sobre los elementos correspondientes: palabras en el caso de que se traten listas y caracteres en el caso de tratar palabras.

La instrucción del ejemplo aplica el operador BUTLAST al dato de entrada constituido por la lista definida (MI NOMBRE ES); el dato de salida será, en consecuencia, la lista MI NOMBRE. A su vez, esta lista actúa como dato de entrada de un operador FIRST, con lo que el nuevo dato de salida será la palabra MI. Por último, ésta constituye el dato de entrada del primer operador FIRST. Ahora, ya no es una lista el dato afectado por FIRST sino una palabra, de ahí que el resultado coincida con el primer elemento o carácter de la misma: M.

Dos nuevos operadores LOGO destinados al tratamiento de listas (en esta ocasión no son adecuados para operar con palabras), son FPUT y LPUT. Su cometido, ilustrado por el siguiente ejemplo, es añadir un elemento al principio o al final de la lista indicada.

```
PRINT FPUT "JUAN :LISTA1
JUAN MI NOMBRE ES

PRINT LPUT [JUAN PEREZ] :LISTA1
MI NOMBRE ES JUAN PEREZ
```

En el primer caso se inserta la palabra JUAN delante de la LISTA1, por medio del operador FPUT. La segunda instrucción, que ilustra la actuación de LPUT, añade la lista JUAN PEREZ al final de la indicada por medio de la variable LIST1.

Así como FPUT y LPUT sólo admiten listas como segunda entrada, el operador WORD admite únicamente palabras. Su misión es transformar a un conjunto de palabras en una palabra única. Así, por ejemplo, la salida de WORD "POR "QUE es la palabra PORQUE.

Si el número de entradas es superior a dos, es necesario encerrar entre paréntesis al operador WORD y a los datos de entrada que lo acompañan:

(WORD "TA "TE "TI "TO "TU)

Cuando el objetivo sea construir una lista, hay que optar por el empleo de LIST. Este *operador* construye una lista integrada por sus datos de entrada.

Al contrario de WORD, LIST sólo admite dos entradas; éstas pueden ser tanto palabras como listas.

TABLA DE ORDENES – LOGO

Instrucción	Cometido	Operador/Comando
COUNT <objeto>	Cuenta elementos	Operador
FIRST <objeto>	Da el primer elemento	Operador
LAST <objeto>	Da el último elemento	Operador
BUTFIRST <objeto>	Elimina primer elemento	Operador
BUTLAST <objeto>	Elimina último elemento	Operador
LPUT	Añade al final	Operador
FPUT	Añade al principio	Operador
LIST	Construye una lista	Operador
SE	Construye una lista de palabras	Operador
WORD	Construye palabras	Operador
ASCII <palabra>	Da el código ASCII de la inicial	Operador
CHAR <número>	Da el caracter ASCII correspondiente	Operador

<objeto>: puede ser una palabra, una lista, un número o una variable (ver tabla de entradas y salidas).

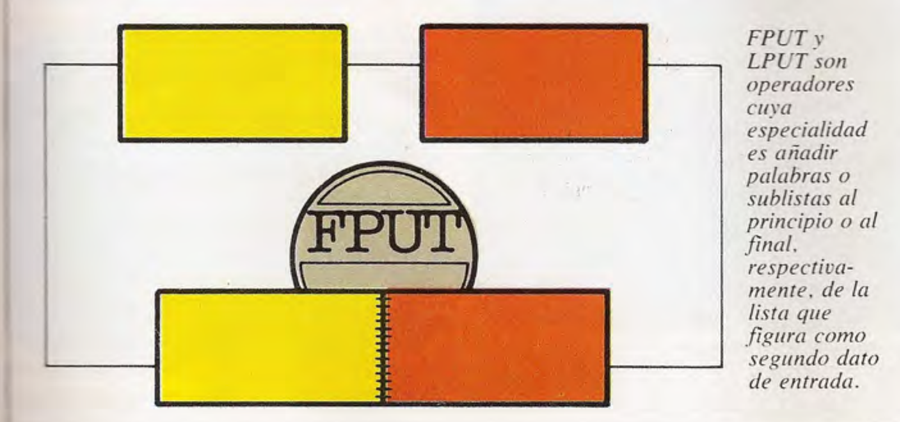
```
MAKE "YO [JUAN PEREZ]
PRINT WORD "JUAN "PEREZ
JUANPEREZ

PRINT LIST :LISTA1 :YO
[MI NOMBRE ES] [JUAN PEREZ]
```

El ejemplo añade a la definición realizada anteriormente (LISTA1), la definición de una nueva variable de tipo lista con la que operar (YO). La segunda instrucción PRINT incluye al operador LIST, cuyo efecto es construir una nueva lista a partir de las dos sublistas especificadas como dato de entrada.

Otro de los operadores de esta categoría es SE (SEntence). Admite cualquier número de entradas, expresándolas al igual que en el caso de WORD.

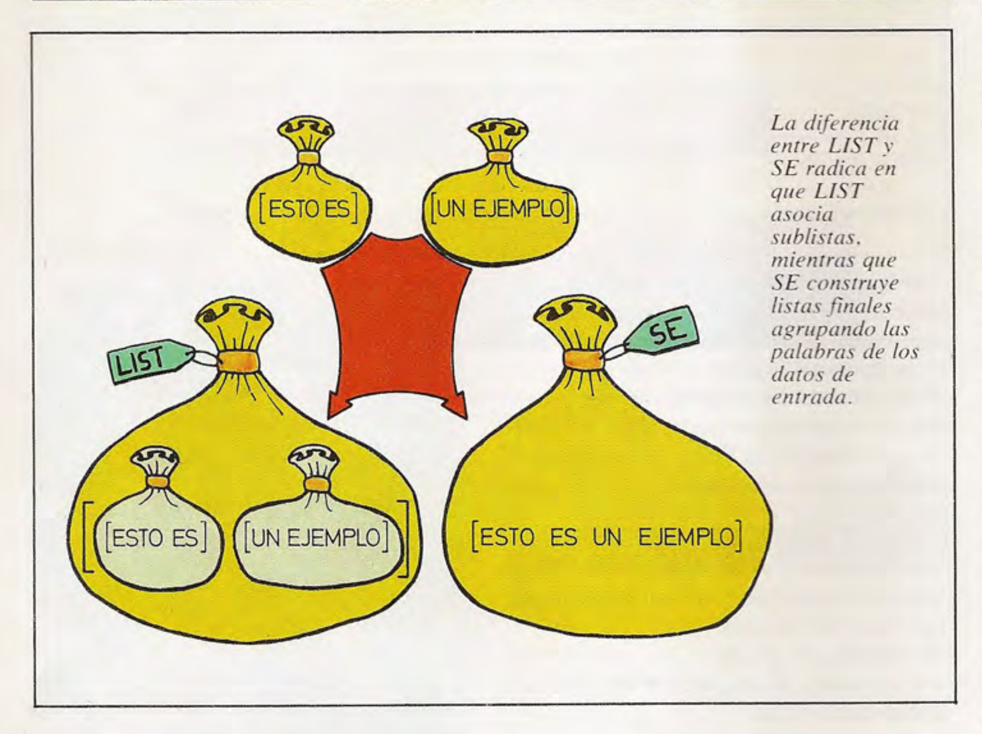
SE sintetiza una lista resultante a partir de sus entradas, si éstas son palabras, o a partir de los elementos de los datos de entrada si éstos son listas. En definitiva, la lista de salida de SE constará de palabras y nunca de sublistas como era el caso de LIST.



FPUT y LPUT son operadores cuya especialidad es añadir palabras o sublistas al principio o al final, respectivamente, de la lista que figura como segundo dato de entrada.

NATURALEZA DE LOS DATOS DE ENTRADA Y SALIDA

Operador	Entrada1	Entrada2	Salida
COUNT	<pal/lis>		<número>
FIRST/LAST	<palabra>		<caracter>
	<lista>		<palabra/sublista>
BUTFIRST/BUTLAST	<palabra>		<palabra>
	<lista>		<lista>
LPUT/FPUT	<pal/lis>	<lista>	<lista>
WORD	<palabra>	<palabra>	<palabra>
LIST	<pal/lis>	<pal/lis>	<lista>
SE	<pal/lis>	<pal/lis>	<lista>
ASCII	<caracter>		<número>
CHAR	<número>		<caracter>



La diferencia entre LIST y SE radica en que LIST asocia sublistas, mientras que SE construye listas finales agrupando las palabras de los datos de entrada.

Una prueba palpable de tal distinción entre SE y LIST la aporta la inclusión del operador COUNT en las dos últimas instrucciones PRINT del siguiente ejemplo:

```
PRINT SE :LISTA1 :YO
MI NOMBRE ES JUAN PEREZ

PRINT COUNT LIST :LISTA1 :YO
2

PRINT COUNT SE :LISTA1 :YO
5
```

El primer COUNT aplicado a LIST cuenta las sublistas (2), mientras que el segundo cuenta las palabras resultantes (5).

Los párrafos precedentes resaltan la importancia que tiene distinguir perfectamente en las entradas y salidas cuándo se trata de palabras y cuándo de listas.

Si un operador recibe una entrada de tipo erróneo, el ordenador mostrará en la pantalla un mensaje de error. Por ejemplo, después de teclear: FPUT "IN "CREIBLE. Aparecerá el mensaje: FPUT DOESN 'T LIKE "CREIBLE AS INPUT (FPUT no admite "CREIBLE como entrada). Un ligero repaso al operador FPUT le hará reparar en que la segunda entrada no puede ser una palabra sino que debe ser una lista.

EL OPERADOR ASCII

El ordenador almacena y trata la información codificada a modo de palabras binarias o grupos de ceros y unos. Tanto las instrucciones como los datos (numéricos o alfanuméricos) adoptan este aspecto para que puedan ser manipulados por la máquina.

A la hora de proceder a la codificación, el ordenador puede acogerse a uno de los muchos códigos alfanuméricos, estandarizados o no, capaces de adecuar la información para que sea manipulable por los circuitos electrónicos. Entre ellos, el más estendido es el código ASCII (American Standard Code for Information Interchange). A pesar de su naturaleza de código estándar, existen distintas versiones del código ASCII, cuyas diferencias radican en la inclusión de un mayor o menor número de caracteres especiales; no obstante, las letras tienen siempre asignados los mismos códigos: del 65 (A) al 90 (Z). El LOGO dispone de operadores que permiten la conversión de carácter o código y viceversa. El primero de ellos, ASCII, devuelve el código del carácter que se especifique como dato de entrada.

La función opuesta corre a cargo del operador CHAR. Este transforma un número de entrada en el carácter ASCII correspondiente.

```
PRINT ASCII "A
65
PRINT CHAR 66
B
PRINT CHAR SUM 1 ASSCII "B
C
```

Los ejemplos anteriores ilustran la actuación de ambos operadores.

Estructura del CP/M

La arquitectura interna del sistema operativo

Al conectar el ordenador, la pantalla se ve ocupada por un mensaje descriptivo de la versión CP/M utilizada. Este simple hecho conlleva toda una frenética actividad en el interior de la máquina. Una sucesión de rápidas operaciones que pasan completamente desapercibidas para el usuario, a no ser por los siseos y crujidos que proceden de la unidad de disco, reveladores de que algo está sucediendo en la intimidad del ordenador.

¿QUE OCURRE EN LA INTIMIDAD DE LA MAQUINA?

La descripción cronológica de los hechos es en breve síntesis la que se describe a continuación. Un programa "cargador" de arranque en frío (cold start), viaja desde el disco hacia la memoria residente del sistema. La función de este programa no es otra que gestionar la carga del sistema operativo desde el disco rígido o disquete a la memoria interna, cediendo el control, a continuación, al propio sistema operativo CP/M. Este se encargará de ultimar las operaciones de inicialización y de visualizar en la pantalla el mensaje de presentación, acompañado del caracter de petición o "prompt" de sistema.
Toda la actividad da comienzo con la carga en memoria de un programa (cold start) cuya única misión es gestionar la carga de otro programa: el sistema operativo. ¿Cuál es la razón de esta aparente duplicidad? La respuesta se encuentra en la necesidad de garantizar la independencia entre el sistema operativo y la máquina, y ello es, precisamente, lo que se consigue con la intervención previa del programa cargador. De esta forma, es posible que microordenadores completamente diferentes puedan utilizar las mismas unidades de discos y la misma copia del sistema operativo CP/M.

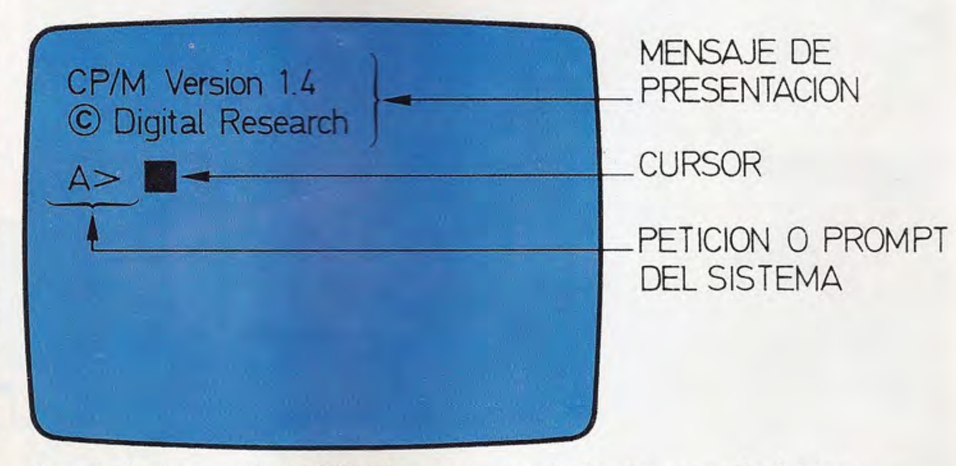
ORGANIZACION INTERNA

La gestión de las diversas funciones del sistema operativo no corren a cargo de un único programa completo; en realidad, se utilizan pequeños módulos menos complejos y orientados específicamente a resolver una determinada tarea. En definitiva, el sistema operativo está construido a base de un conjunto de módulos especializados. Semejante estructura modular mejora su rendimiento, a la vez que se proporciona una gran flexibilidad ante posibles cambios y modificaciones.

Los distintos módulos que integran el sistema operativo CP/M son los que se indican a continuación:

- CCP: Procesador de comandos de consola o *Console Command Processor*.
- BDOS: Sistema operativo básico de disco o *Basic Disk Operating System*.
- BIOS: Sistema básico de entrada/salida o *Basic Input/Output System*.

La función del primero de ellos (CCP) es facilitar una comunicación directa con la máquina por medio de ciertas frases convenidas, que sirven para ejecutar órdenes o comandos. El segundo módulo (BDOS) está especializado en la gestión de la memoria secundaria, o dispositivos para el almacenamiento masivo y permanente de información; desde un nivel lógico, sin entrar en la parte física de estos dispositivos. Por último, el módulo (BIOS) es el responsable de la ejecución de las funciones dependientes del soporte físico o hardware del sistema.
La unión conjugada del CCP con el BDOS,



Pantalla de presentación del CP/M. Una vez conectado el equipo y cargado el sistema operativo, la pantalla se ve ocupada por un mensaje de presentación. Bajo éste, aparece el indicador de petición o «prompt» del sistema, acompañado por el cursor que señala el punto de escritura.

constituye la zona lógica del sistema, independiente del entorno y, por lo tanto, transportable a cualquier otro sistema, aun cuando éste no comparta la misma configuración de dispositivos periféricos. Por contra, el módulo BIOS contiene los programas que establecen la comunicación directa con los dispositivos físicos.
Con frecuencia, el BIOS es escrito por el propio fabricante del ordenador y, debido a ello, no es trasladable a otro entorno físico (hardware) distinto al que ha condicionado su diseño.

CONVENIOS ADOPTADOS

Para una visión más clara de la notación utilizada, tanto en el texto como en las figuras, se convendrán los siguientes puntos:

— En las referencias a comandos, la zona tecleada por el usuario aparecerá en letra cursiva.

— Los caracteres especiales, obtenidos al accionar simultáneamente la tecla CONTROL (representada por CTRL, CTL o ATL en algunos microordenadores) y otro caracter, se representarán por medio del símbolo CTRL seguido por un guión y el caracter correspondiente, todo ello encerrado entre corchetes angulares. Por ejemplo, el caracter de control resultante de pulsar la tecla CONTROL y la letra U, se representará en la forma: <CTRL-U>.

— Al hablar de disco en general, no se harán distinciones entre el disco rígido y el flexible o disquete.

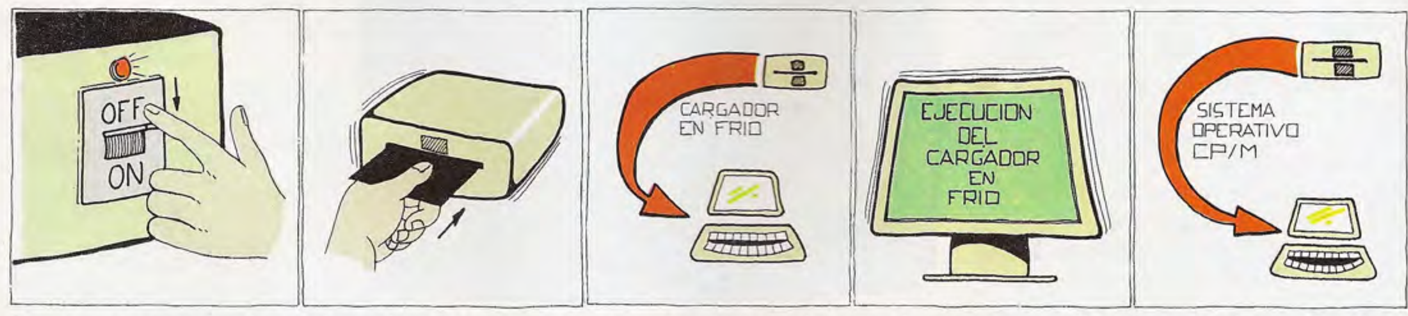
— El retorno de carro <CR> necesario para la ejecución de los comandos, no se indicará a no ser que se considere ineludible su presencia para una mejor comprensión.



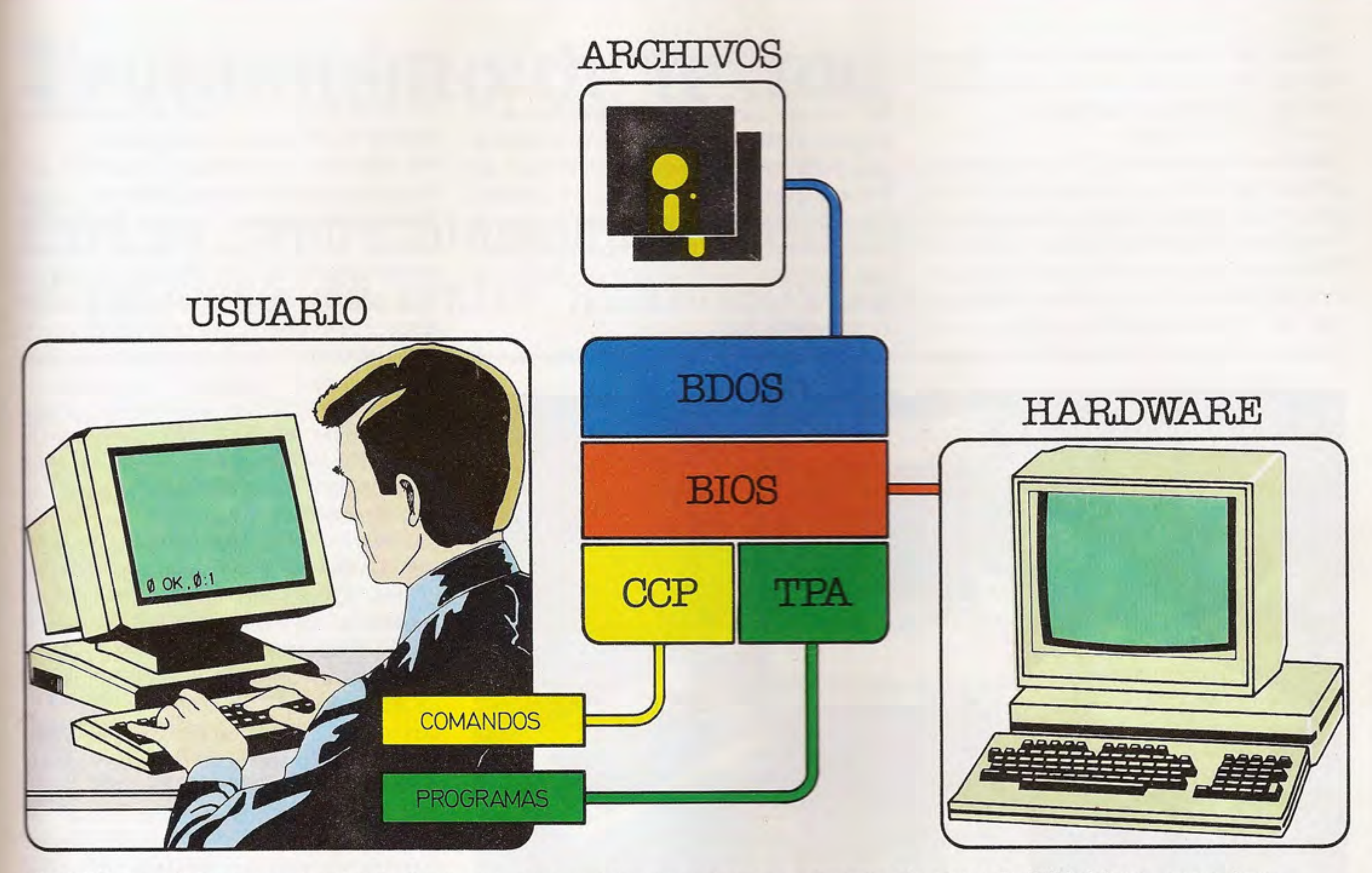
El CP/M suele residir en un disco, ya sea de tipo flexible o rígido. En el primer caso, el usuario debe introducir el disco en la correspondiente unidad para que el ordenador pueda acceder a la inteligencia básica que aporta el sistema operativo.

LA COMUNICACION HOMBRE-MAQUINA

El diálogo entre el usuario y el microordenador, se lleva a efecto a través de un repertorio de órdenes o comandos, cuyo número es limitado y que deben ajustarse a unas reglas de vocabulario y sintaxis.
El módulo CCP es, esencialmente, un intérprete de comandos: acepta las órdenes introducidas por el usuario a través del teclado y las analiza sintácticamente para verificar su corrección antes de ejecutarlas. Tal ejecución puede involucrar a los otros dos módulos, BDOS y BIOS, en el caso de que la puesta en práctica del comando en cuestión exigiera la intervención de alguno de ellos.
La disponibilidad del sistema para aceptar un comando viene determinada por la aparición en la pantalla del indicador de presencia o de petición de entrada: el *prompt* del sistema. Este consta de una letra indicativa del disco accesible en el preciso instante y del caracter "mayor que" (>). Habitualmente, aparece como "A>".
Al introducir cualquier comando, el CCP lo analiza y comprueba que es correcto. Aunque su expresión sea correcta, si forma parte de una lista de comandos que residen permanentemente en la memoria del microordenador, se buscará en el directorio del disco un fichero de tipo "COM" cuyo nombre comparta los ocho primeros caracteres (o hasta el primer espacio en la línea del comando). Una vez localizado, el fichero se carga en la memoria y, a continuación, se ejecuta. Este tipo de comandos son los denominados *comandos transitorios*.
Puede ocurrir que un programa de usuario sea excesivamente voluminoso e invada la zona reservada al CCP; en tal caso, se perderá la comunicación con el sistema, y el usuario perderá la facultad de dar órdenes inteligibles para el microordenador. Para romper con esta situación, puede recurrirse a la combinación de dos teclas <CTRL-C>, que accionadas simultánea-



Tras conectar el ordenador, se desarrollan toda una serie de operaciones que concluyen con la puesta en pantalla del mensaje de presentación y, en definitiva, con el trasvase al interior de la máquina de su inteligencia elemental: el sistema operativo. Después de dar tensión al ordenador e introducir el disco del sistema operativo en el periférico correspondiente, la máquina lee del disco un programa "cargador en frío", cuya ejecución gestionará la carga del sistema operativo.



El CP/M es un sistema operativo dotado de una estructura modular que mejora su rendimiento y le otorga flexibilidad frente a posteriores cambios o actualizaciones. Cada uno de los módulos gestiona un determinado grupo de tareas propias del ordenador.

mente provocan el abandono de cualquier programa en curso de ejecución, y reinicializar el sistema: vuelven a cargar los diversos módulos que componen el CP/M. Esta operación recibe el nombre de *arranque en caliente* o "Warm start".

ASPECTO DE UN COMANDO

Cualquier comando, ya sea del sistema o del usuario, adopta la forma de conjunto de palabras o cadenas de caracteres alfanuméricos, separadas por espacios en blanco, y que terminan obligatoriamente con un retroceso de carro (acción sobre la tecla RETURN). La primera palabra, localizada a continuación del "prompt" del sistema, representa al comando propia-

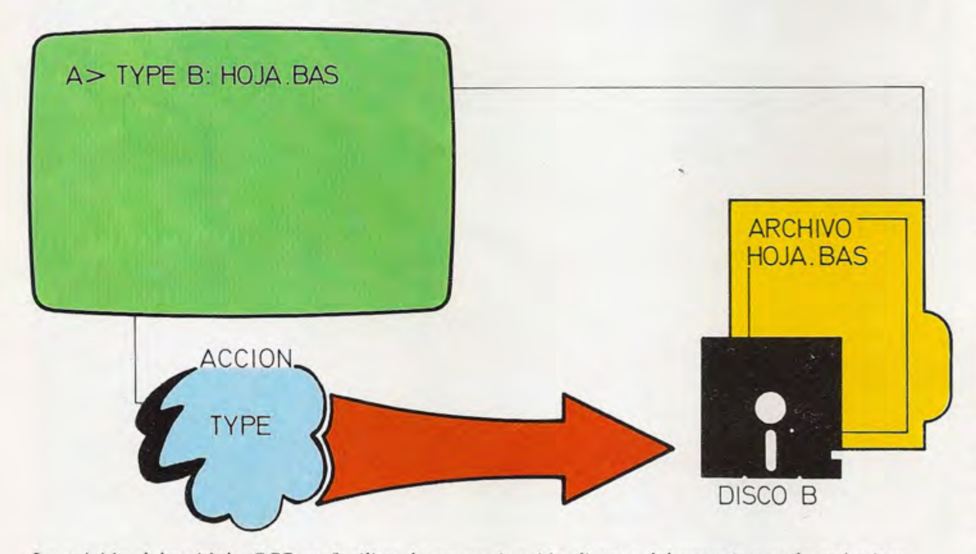
Control directo desde el teclado

Durante la introducción de los comandos pueden cometerse errores; de ahí que sea imprescindible disponer de una herramienta que permita modificar los comandos incorrectos antes de concluirlos con la orden del retroceso de carro (CR). Tal posibilidad la brindan los diversos caracteres de control que apoyan la edición de comandos.
Estos caracteres especiales son combinaciones de la tecla CONTROL con ciertas teclas alfabéticas (letras), y permiten al usuario un control limitado de las operaciones que realice en el terminal. La tabla adjunta relaciona algunos de los caracteres de control más frecuentes.
Ciertos caracteres de control son aceptados por el sistema en cualquier momento —por ejemplo: <CTRL-H>, <CTRL-C>, <CTRL-S>...—, no sólo en el intervalo de tiempo que media entre el comienzo de la entrada del comando y la pulsación de <CR>, que pone fin a la introducción de la orden en la máquina, activando su ejecución.

CARACTERES DE CONTROL DEL CP/M	
<CTRL-H>	Borra el caracter anterior al cursor.
<CTRL-U>	Borra una línea completa.
<CTRL-M>	Equivalente a <CR>.
<CTRL-E>	Indica que el comando continúa en la línea siguiente.
<CTRL-S>	Congela la impresión sobre la pantalla.
<CTRL-Z>	Indica fin del fichero.
<CTRL-C>	Carga el sistema operativo por medio de un arranque en caliente.

mente dicho; las palabras que lo siguen son los argumentos o parámetros asociados al comando. Por ejemplo:
A > *ED TEXTO.BAS*
contiene el comando ED, que ordena la edición de un fichero, completado con el argumento TEXTO.BAS (nombre del fichero sobre el que debe actuar el editor).
Pocos son los comandos que no actúan directamente sobre un fichero; debido a ello, es preciso contar con una buena normalización que permita hacer referencia a un fichero eliminando toda posible ambigüedad.
El formato completo que identifica a un fichero consta de una referencia al disco al que pertenece, seguida por el nombre del fichero y el tipo de fichero. El caracter "dos puntos" (:) se utiliza para separar la referencia al disco del nombre del fichero y el "punto" (.) para separar el nombre del tipo. Por tanto, la referencia:
B:NOMBRE.BAS
identifica a un fichero contenido en el disco B, denominado NOMBRE y que pertenece a un tipo de ficheros (BAS) cuya característica es que son programas fuente escritos en lenguaje BASIC.
No siempre es necesario describir exhaustivamente el fichero utilizado, ya que el sistema operativo toma por defecto los valores no definidos, e incluso permite omitir alguno de los identificadores sin que por ello se genere un mensaje de error. Por lo que respecta al disco, el sistema operativo toma por defecto el disco que aparece en la petición del sistema.
El nombre del fichero siempre es necesario definirlo, aunque algunos sistemas operativos permiten referenciar al fichero cuyo nombre y tipo son nulos. A su vez, el tipo puede omitirse siempre que no sea exigible (tal es el caso de los compiladores); en ciertas ocasiones puede tomarse el tipo por defecto, como, por ejemplo, cuando se hace referencia a un comando no residente o transitorio y no se especifica que el tipo debe ser ejecutable (.COM).
Los identificadores de las diversas unidades de disco disponibles coinciden con las primeras letras del alfabeto. La primera unidad, utilizada generalmente como "almacén" externo del propio sistema operativo, se nombra con la letra "A"; éste corresponde al disco número "0". La letra "B" designa al disco cuyo número es el "1", y así sucesivamente, pudiendo llegar hasta el disco número "15", al que corresponderá la letra "P".
El empleo de este método de identificación por parte de los distintos fabricantes no está del todo normalizado. Hay fabricantes que reservan las letras finales del alfabeto ("M", "N", "O" o "P") para designar a las unidades de disco rígido y destinan los primeros caracteres alfabéticos ("A", "B", "C", o "D") a discos flexibles.
El cambio de disco sobre el que se actúa se consigue fácilmente sin más que indicar, detrás del "prompt" del sistema, la referencia del nuevo disco al que se desea acceder seguido por el signo "dos puntos". Por ejemplo:
A> C :
A raíz de la ejecución de esta orden, se modificará el "prompt" o indicador de presencia del sistema. Este reflejará el cambio de disco ordenado:
C>
A partir de este instante, el disco C será considerado como disco de trabajo hasta que el sistema operativo reciba una indicación en contra.

El módulo BIOS es el brazo ejecutor de las órdenes que dirige la máquina a la zona física de los periféricos que dependen de su control.



La misión del módulo CCP es facilitar la comunicación directa del usuario con la máquina: una comunicación que se establece por medio de las órdenes o comandos.
Un comando consta de dos zonas fundamentales: la primera expresa la acción a realizar (por ejemplo TYPE), mientras que la segunda especifica el objeto o elemento destinatario de la acción (archivo HOJA, de tipo BAS y residente en el disco B).

Tratamiento de textos

Qué es, cómo funciona y qué ventajas aporta el proceso de textos automatizado

El tratamiento manual de palabras para la edición de cartas, documentos, informes, contratos o cualquier otro tipo de texto, es un proceso caro; tanto por el tiempo a invertir en la producción del texto escrito como por su importe económico. En el mundo profesional y empresarial, este coste se ve multiplicado por la vorágine informativa que ha hecho aumentar de forma vertiginosa el número de originales y copias a manejar.

Como en otros muchos campos de actividad, la solución más rentable y eficaz para reducir tiempos y coste ha sido mecanizar el tratamiento de textos. En muy pocos años, esta actividad ha crecido tan espectacularmente que, hoy en día, se pueden encontrar infinidad de sistemas, basados en distintos equipos informáticos, especializados en este tipo de procesos.

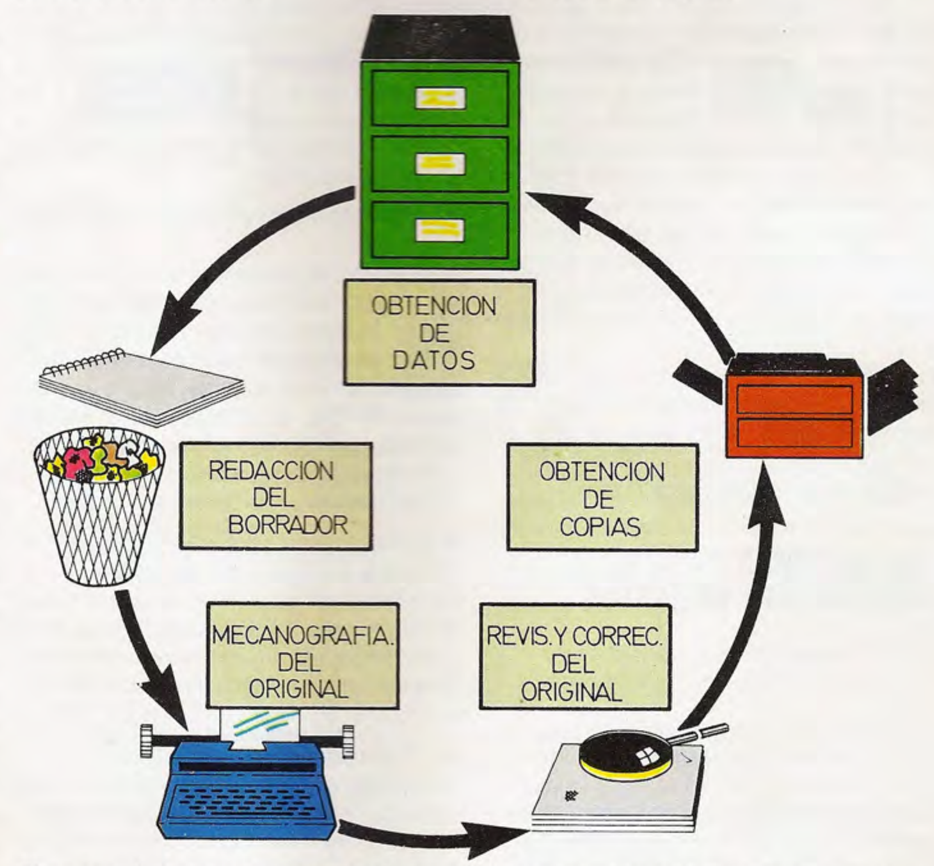
¿EN QUE CONSISTE EL TRATAMIENTO de TEXTOS?

Las operaciones manuales necesarias para producir cualquier texto escrito, antes de la llegada de la solución informática, eran las siguientes:

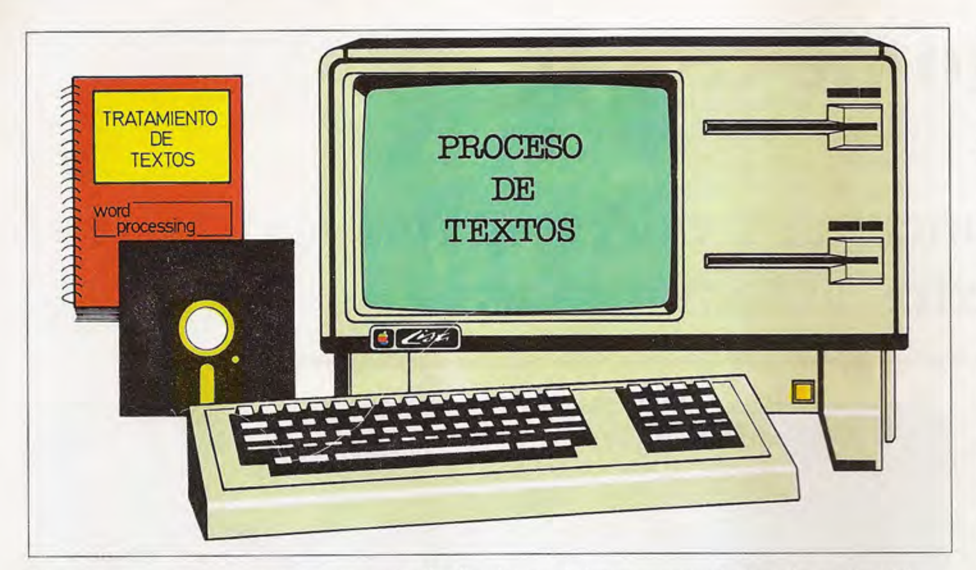
1. *Obtención de los datos* necesarios para producir el texto. Para ello se utilizaban archivadores, ficheros, etc.
2. *Redacción del borrador.* Una vez recopilada la información, se escribía un manuscrito redactado como primera aproximación del texto final.
3. *Producción del original* mecanografiando el borrador manuscrito.
4. *Revisión y corrección* de errores mecanográficos.
5. *Obtención de copias* del escrito original.
6. *Archivo de una de las copias,* por si en un futuro resultara necesaria para obtener información al respecto.

En la actualidad, todas estas operaciones se pueden realizar con el apoyo de un sistema informático. Veamos cómo se efectuarían las seis operaciones si se opta por el empleo de un ordenador capacitado para el tratamiento de textos:

1. *Obtención de datos.* En los ficheros mecanizados del ordenador se encontrará la mayor parte de la información necesaria para producir el escrito.
2. *Redacción del borrador.* Utilizando un formato en el que no se considera aún la estética de la presentación, puede elaborarse la información recopilada para redactar, fácil y rápidamente, un borrador del texto.
3. *Producción del original.* En este punto será el propio sistema mecanizado el que a partir del borrador y varios parámetros definitorios del formato (tipo de letra, líneas por página, número de caracteres, paginación automática, etc.) se encargará de preparar el original.



El proceso de textos no mecanizado obliga a realizar toda una serie de operaciones manuales que encarecen la actividad y, a su vez, dilatan el tiempo que transcurre entre el instante en el que se desea el documento y su obtención final.



Con el tratamiento de textos automatizado disminuye el tiempo necesario para la producción de documentos escritos, mejora la calidad de los mismos y se reduce el coste económico.

4. *Revisión y corrección.* Antes de imprimir el texto, se puede observar en una pantalla como aparecerá escrito sobre el papel; en este instante pueden realizarse las oportunas correcciones.
5. *Obtención de copias.* Es importante señalar que hasta este momento no ha sido necesario editar ningún papel escrito. Ahora, una vez garantizada la corrección del texto, pueden ya obtenerse automáticamente todas las copias que se deseen.
6. *Archivo de una de las copias.* Esta es una tarea superflua, puesto que el texto permanecerá almacenado en la memoria de masa del sistema. Un hecho que supone una sustancial economía en cuanto a volumen de archivo y manipulación de copias.

En definitiva, no cabe duda de que el empleo de un sistema para el tratamiento de textos aporta, además de un ahorro de tiempo y dinero, una clara mejora en la calidad de los escritos producidos.

LOS SISTEMAS PARA EL TRATAMIENTO DE TEXTOS

Como ya apuntábamos al principio, existen muchos y muy distintos sistemas para el tratamiento de textos. A continuación se relacionan algunos de los más importantes:

- *Máquinas de escribir electrónicas*

Son los sistemas más elementales para el proceso de textos. Básicamente, son máquinas de escribir tradicionales a las que se acopla una pequeña memoria y algunas teclas adicionales que introducen ciertas funciones: justificación del texto entre ambos márgenes, centrado automático, alineación por la izquierda o derecha... En algunos casos también incorporan un visor para comprobar la corrección del texto antes de imprimirlo.

- *Sistemas de máquinas de escribir electrónicas*

Con objeto de aumentar la memoria disponible en cada máquina y hacerla permanente, pueden conectarse una o varias máquinas de escribir a un ordenador. Además de las funciones de edición propias de la máquina, el ordenador aportará una pantalla para la visualización de los textos y facilitará la comunicación entre los distintos "productores de textos".

- *Ordenadores personales con programas para el tratamiento de textos*

Sin lugar a dudas, éste es el sistema para el tratamiento de textos más frecuente y popularizado. Como su propio nombre indica, se basa en la asociación de dos elementos:

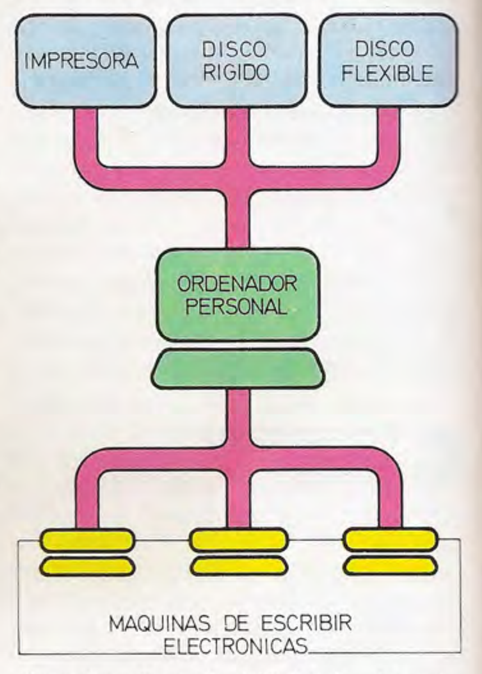
— *Ordenador personal*

Un equipo de moderado tamaño y precio útil para realizar los trabajos que tradicionalmente se reservaban a los grandes ordenadores. Los dos usuarios típicos de ordenadores personales son los profesionales independientes y las pequeñas o medianas empresas. Ambos usuarios suelen tener necesidad de producir numerosos escritos y pueden obtener notables ventajas si deciden utilizar el ordenador personal para el tratamiento de textos.

— *Programa (software) para el tratamiento de textos*

Como ya anticipábamos, el ordenador personal puede utilizarse para resolver problemas de muy diversa índole. Para cada uno de ellos, hay que disponer de uno o más programas que serán ejecutados en el ordenador tantas veces como desee el usuario. Si se opta por utilizar el ordenador personal para realizar el tratamiento de textos, será imprescindible contar con el software de aplicación adecuado.

En esta misma sección estudiaremos, más adelante, los principales paquetes de aplicación destinados al procesamiento de textos en ordenadores personales.



Para elevar la productividad de los sistemas basados en máquinas de escribir electrónicas, éstas pueden conectarse a un ordenador personal de forma que compartan los recursos del mismo.

- *Procesadores de textos dedicados*

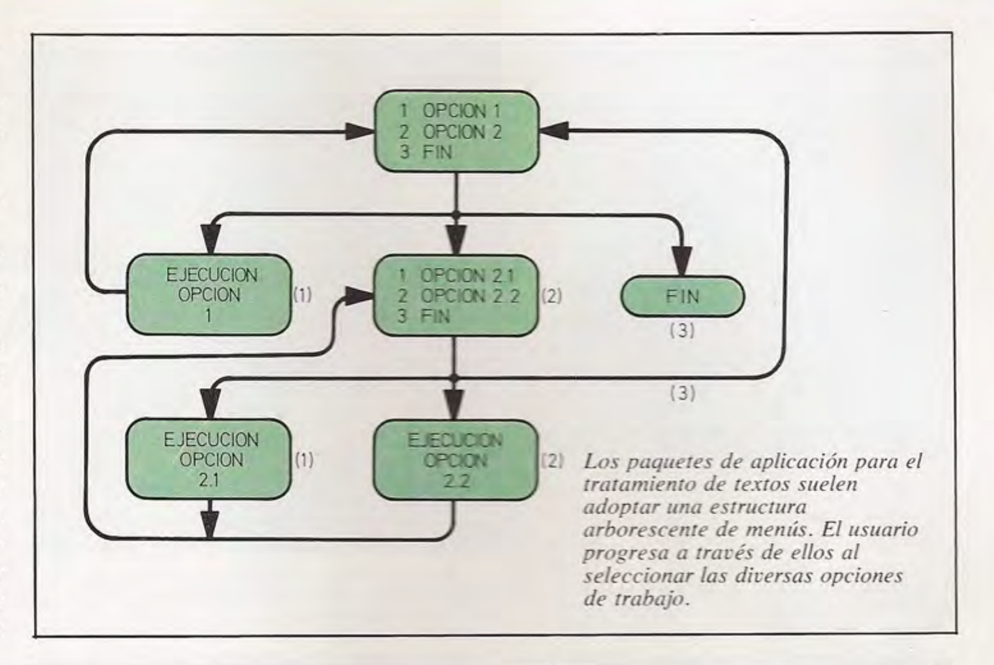
En algunos casos, para reducir el precio del hardware (equipos) y del software (programas) se opta por microprocesadores especializados exclusivamente en el tratamiento de textos. La adquisición de un equipo específicamente diseñado para tal actividad, impide utilizar el hardware para resolver otro tipo de problemas. Evi-

dentemente, esta solución sólo es recomendable para empresas con un gran volumen de textos a tratar y dotadas de otros procedimientos de mecanización general.

• *Ordenadores con programas para el tratamiento de textos*

Al igual que puede utilizarse un ordenador personal "instruido" por programas para el tratamiento de textos, también es posible utilizar programas de esta índole con otros equipos, como mini-ordenadores e incluso grandes ordenadores.

Normalmente, el empleo de grandes ordenadores para el tratamiento de palabras suele estar reservado a la documentación de programas informáticos, ya que resulta muy caro dedicar equipos grandes al tratamiento de textos. Por ello, lo habitual es que las empresas recurran a alguno de los sistemas descritos anteriormente, aun en

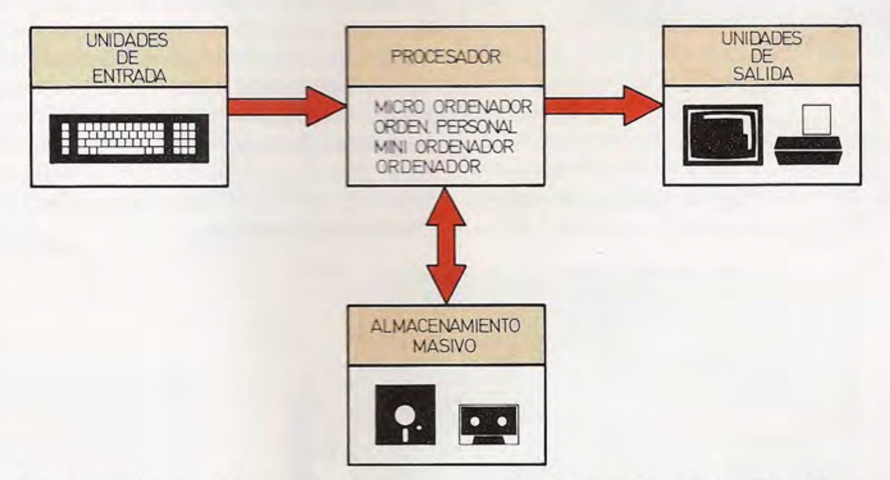


Los paquetes de aplicación para el tratamiento de textos suelen adoptar una estructura arborescente de menús. El usuario progresa a través de ellos al seleccionar las diversas opciones de trabajo.

Elementos hardware para el proceso de textos

Para realizar un tratamiento de textos automatizado, resulta imprescindible la intervención de un procesador. Este es el elemento básico para gestión de todo el conjunto de tareas cuya culminación es la edición de un texto. No obstante, sin la participación de al menos un periférico dedicado a la entrada del texto original, y otro para la salida del texto ya procesado, el ordenador sería incapaz de comunicarse con el usuario. En definitiva, las exigencias hardware hacen imprescindible la presencia de un procesador (unidad central del ordenador), una unidad de entrada y una unidad de salida.

Con esta configuración mínima es posible almacenar el texto introducido en la memoria del ordenador y, una vez tratado, reproducirlo a través de una impresora (unidad de salida). No obstante, si una vez finalizada la sesión se desea obtener una nueva copia de un texto introducido en alguna sesión anterior, será necesario volver a introducir el texto en su totalidad. Para solucionar este inconveniente, es preciso disponer de una unidad para el almacenamiento permanente de la información (los textos en el caso que nos ocupa). Este cometido queda encomendado a los periféricos de almacenamiento o memoria de masa asociadas al ordenador. Su presencia permite ya disponer de una configuración completa, con los elementos hardware típicos e imprescindibles para realizar con propiedad el tratamiento de textos.



Esta breve síntesis, cabe completarla citando algunas de las variantes que pueden adoptar los distintos elementos hardware del sistema.

1. *Procesador*

Puede utilizarse desde un ordenador personal de cualquier categoría, hasta un ordenador de gran tamaño; si bien, hay que constatar que corresponde un mayor protagonismo a los ordenadores personales.

2. *Unidades de entrada/salida*

Tanto para introducir los textos originales, como para comunicar al usuario los resultados provisionales del tratamiento, suele utilizarse un terminal (teclado más pantalla). La entrada se realiza a través del teclado, mientras que la salida adopta la forma de presentación en pantalla.

El periférico de salida adecuado para editar el texto definitivo es la impresora de papel.

3. *Unidades para el almacenamiento permanente*

En este punto son varias las alternativas: discos magnéticos de tipo rígido (muy adecuados por su elevada capacidad y velocidad de acceso para recoger datos de forma permanente), los tradicionales discos flexibles, o incluso simples casetes convencionales.

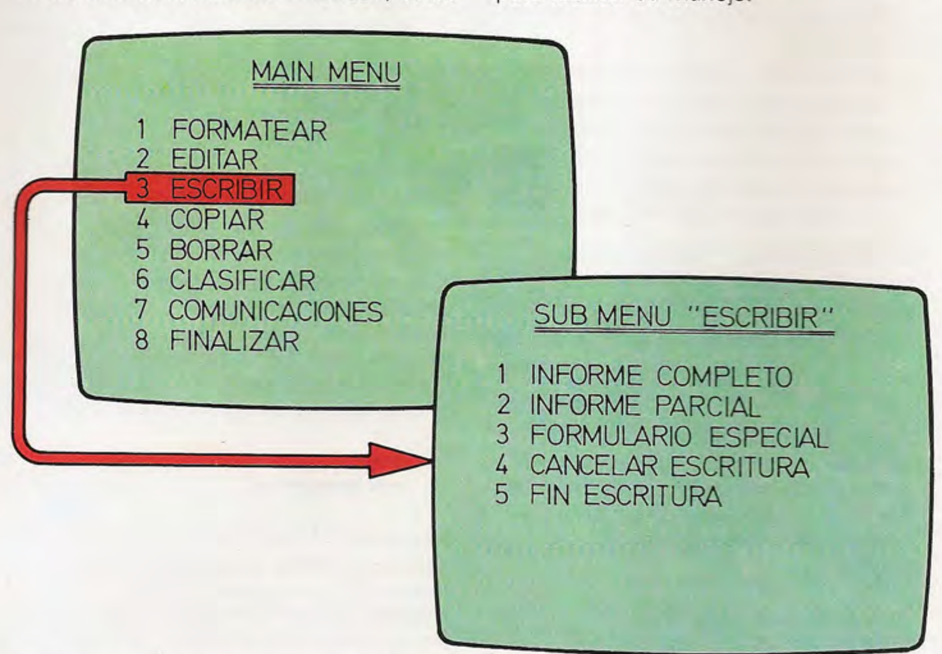


Los modernos ordenadores personales han irrumpido en el ámbito del tratamiento de textos, aportando comodidad y eficacia respecto a los medios tradicionales.

el caso de que ya dispongan de un gran centro de proceso de datos.

¿COMO FUNCIONA UN SISTEMA PARA EL TRATAMIENTO DE TEXTOS?

El método habitual se plasma en una estructura arborescente de «menús»; éstos permiten al usuario ir tomando las decisiones que van a marcar el funcionamiento del programa. Cada "menú" consiste en un formato, visualizado en la pantalla, a través del que el sistema ofrece al operador un abanico de posibilidades para que decida cuál de ellas desea ejecutar. Opcionalmente cada "menú" puede incorporar una explicación de los resultados que se producirán según la opción elegida; en este último caso, se dice que el sistema dispone de TUTOR o de apoyo "HELP" para facilitar su manejo.



El menú principal (main menu) ofrece las opciones generales que sintetizan las posibilidades del tratamiento de textos. Cada una de sus opciones da paso al correspondiente submenú cuya finalidad es precisar la actividad elegida.

El "menú" principal es el que aparece inmediatamente después de invocar o "llamar" al sistema de tratamiento de textos; con frecuencia, suele recibir el nombre de "MAIN MENU". Al elegir alguna de sus opciones, debe aparecer en pantalla un nuevo menú, al que se denomina "SUB-MENU" (menú secundario), que ofrece nuevas posibilidades dentro de la opción seleccionada.

En cualquiera de los menús ya citados (HELP MENU, MAIN MENU o SUBMENUS), existe una última línea de comunicación a la que se denomina PROMPT. A través de ella el sistema se comunica con el usuario. Por ejemplo, si en un submenú existen las opciones A, B, C y D y el usuario pulsa la letra E, que no corresponde a ninguna de las opciones existentes, en la línea PROMPT aparecerá un mensaje del siguiente estilo: "LA OPCION <E> ES DESCONOCIDA".

También es muy frecuente, cuando se utiliza una opción para la edición de texto por pantalla, la existencia de dos o tres líneas, situadas en la parte superior de la pantalla, en las que se indican las características de la situación en que nos encontramos. En la primera de ellas suele referenciarse la opción en la que se encuentra el sistema, el título del documento, el espacio disponible, etc. Las siguientes, suelen estar ocupadas por una "cabecera" que sirve para identificar la posición del cursor (fila y columna en la que se puede escribir).

UNA BREVE SINTESIS

En este primer capítulo dedicado a los sistemas para el tratamiento de textos se ha contestado sucintamente a tres preguntas de carácter general: ¿En qué consiste el tratamiento mecanizado de textos?, ¿cuáles son los principales tipos de sistemas para el tratamiento de textos? y ¿cómo funciona un sistema para el tratamiento de textos? En próximos capítulos se estudiarán con detalle las principales aplicaciones que existen en el mercado informático para el tratamiento de textos en ordenadores personales.

Edición de programas

Escritura, corrección y puesta a punto de programas BASIC

A lo largo de los capítulos precedentes se han presentado algunos de los comandos que forman parte del vocabulario del lenguaje BASIC. Comandos cuyo objetivo es construir las instrucciones que darán cuerpo a los programas destinados a "educar" al ordenador. En este punto de la obra cabe plantearse un interrogante: ¿cómo hay que escribir los programas de forma que se aprovechen al máximo las posibilidades de edición que brinda el BASIC?

Es evidente que el método utilizado hasta ahora resulta eficaz: escribir las instrucciones deseadas y, al final de cada línea, pulsar la tecla correspondiente al retroceso de carro (RETURN o ENTER). Sin embargo, no cabe duda que semejante procedimiento no es generalizable... ¿Qué sucede si se teclea algo mal? Aquí es donde entra en juego una de las herramientas que brinda el ordenador para facilitar las tareas de programación: el editor. Esta es una zona, habitualmente incorporada al traductor de lenguaje BASIC, que agrupa a un cierto número de funciones para facilitar la escritura y corrección de programas.

Existen varios tipos de editores que ofrecen un mayor o menor número de funciones y ponen en práctica diversos métodos de edición.

Antes de pasar a su estudio, es conveniente dar un repaso a los procedimientos de edición más elementales, compartidos por cualquier ordenador abierto al diálogo en BASIC.

El procedimiento más obvio y elemental para corregir el error cometido en una línea de programa consiste, sencillamente, en escribir de nuevo la línea en cuestión. Al introducir una línea de programa precedida por un número utilizado anteriormente, la nueva línea reemplazará a la anterior; en consecuencia, la nueva línea que incorpora la corrección pertinente, reemplazará a su homóloga errónea.

Este mismo método es perfectamente utilizable para borrar líneas de programa. Si se introduce únicamente el número de línea, el ordenador borrará la línea cuyo número coincida con el introducido. En efecto, la acción equivale a reemplazar la línea anterior por la nueva, desprovista de contenido.

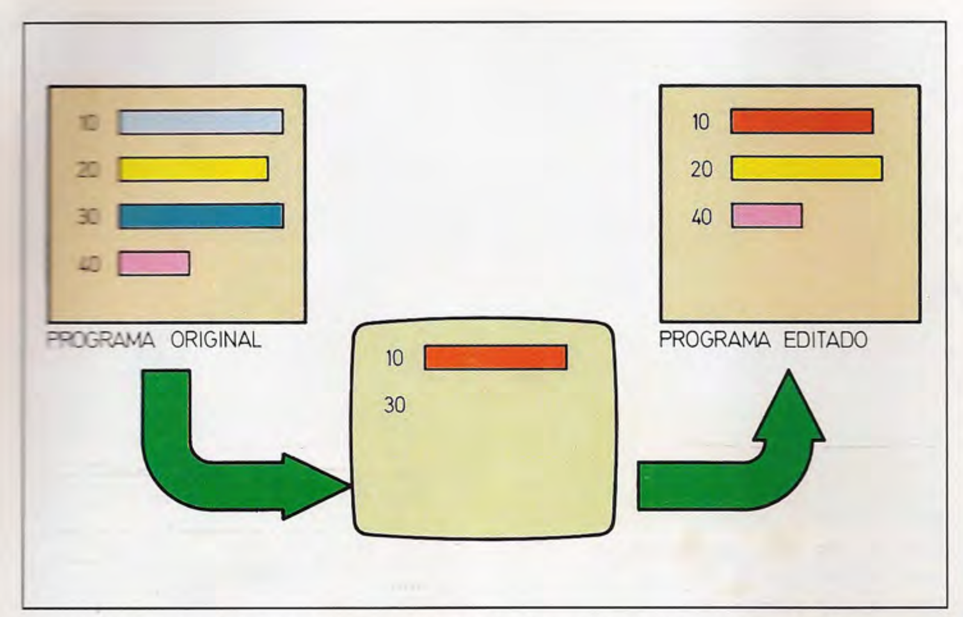
La eficacia de este método elemental es indudable tal como evidencia el siguiente trabajo de edición.

La escritura, corrección y puesta a punto de los programas son tareas englobadas en el apartado de "edición". Esta es una actividad apoyada por el propio traductor de lenguaje BASIC, a través de las denominadas herramientas de edición.

ESCRITURA DE UN PROGRAMA

La sesión de trabajo de un programador empieza, ineludiblemente, con la introducción del programa. Un programa que puede ser tan sencillo como el que aparece en la pantalla:

```
10 REM ENTRADA DE DATOS
20 INPUT "TATO 1"; A
30 PRINT "DATO 1"; A
40 PRINT "EL DATO ES:"; A
■
```



El procedimiento más elemental para la corrección de errores es el que refleja el gráfico adjunto. Para corregir una línea de programa, basta con reescribirla precedida por el mismo número. Si se trata de borrarla, será suficiente con escribir su correspondiente número de línea seguido por una acción sobre la tecla RETURN.

Cuatro líneas de programa cuya introducción se ha realizado ordenadamente y poniendo en práctica la misma secuencia en cada caso: teclear el correspondiente número de línea, su contenido y, por último, darla por concluida con una acción sobre la tecla de retorno (RETURN o ENTER).

EDITOR DE PANTALLA

10
20
30
40
50

DESPLAZAMIENTO DEL CURSOR

Uno de los tres tipos básicos de editores es el de "pantalla". El usuario puede desplazarse a cualquier punto de la misma utilizando las teclas para el desplazamiento del cursor. Una vez colocado el cursor, puede ya introducir las oportunas modificaciones.

El programa está ya en el interior de la máquina y puede ordenarse su ejecución con la orden RUN. Sin embargo, es conveniente comprobar antes su corrección.
Tal como suele ocurrir con frecuencia, se ha deslizado un error en su introducción; un error casi inapreciable (una letra alterada en la línea 20), pero suficiente para inutilizar la eficacia práctica del programa.
¿Cómo corregirlo? Aplicando el procedimiento de edición más elemental, será preciso escribir de nuevo la línea en cuestión, esta vez procurando que no se repita el error.
La línea escrita, cuyo número coincide con el de la que incluía el error, sustituirá a aquella en la memoria del ordenador. Esta circunstancia es fácilmente comprobable, sin más que ordenar la presentación en pantalla del listado por medio del comando LIST:

```
20 INPUT "DATO 1"; A

LIST
10 REM ENTRADA DE DATOS
20 INPUT "DATO 1"; A
30 PRINT "DATO 1"; A
40 PRINT "EL PRIMER DATO ES:"; A
■
```

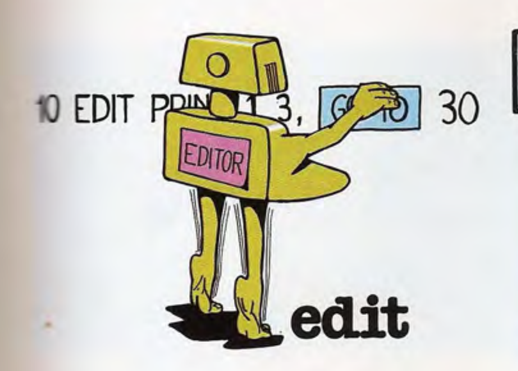
En efecto, el error se ha subsanado: la palabra DATO de la línea 20 aparece ya corregida.
Un nuevo repaso al programa puede decidir al programador por la supresión de la línea 30, cuya presencia es redundante. De nuevo, puede aplicarse el mismo procedimiento: escribir una nueva línea de programa precedida por el mismo número. En este caso, como quiera que se trata de eliminar su presencia, basta con escribir sólo el número de la línea afectada y pulsar la tecla RETURN a continuación. El resultado queda patente al utilizar una vez más el comando LIST:

```
30 (CR)

LIST
10 ENTRADA DE DATOS
20 INPUT "DATO 1"; A
40 PRINT "EL PRIMER DATO ES:"; A
■
```

EDITORES DE PROGRAMAS

Una sesión de trabajo como la relatada en el apartado precedente puede convertirse en una tarea prolongada y tediosa. Normalmente, los programas a editar serán mucho más extensos y sus instrucciones bastante más complejas. En tal caso, es obvio que la repetición del contenido total de las líneas a modificar no será una actividad grata ni eficaz.
Afortunadamente, la mayor parte de los traductores de lenguaje BASIC disponen de herramientas auxiliares para corregir errores y realizar modificaciones con mayor comodidad y rapidez. El conjunto de medios que brinda el ordenador para la edición de programas recibe el nombre de *editor*.
Existen muchos tipos de editores, diferenciados por su potencia y por las posibilidades que ponen a disposición del programador. En todo caso, el aprendizaje del manejo del editor es de suma importancia a la hora de adquirir un ordenador y dispo-



EDIT

Da paso al editor. Permite modificar el contenido de las líneas de programa.

Formato: EDIT [<*número de línea*>]

Ejemplos: EDIT
EDIT 30

nerse a confeccionar programas para el mismo.

En muchos ordenadores está permitido realizar modificaciones y corregir errores (editar) inmediatamente después de haber listado la porción afectada del programa. Normalmente, se puede acceder a la línea deseada accionando las teclas para el desplazamiento del cursor. Una vez posicionado el cursor se procede a modificar la línea, ya sea insertando caracteres o bien realizando una nueva escritura encima de la zona a corregir. Después de realizar los cambios pertinentes, basta con pulsar la tecla RETURN en cualquier punto de la línea para que se haga definitiva la modificación realizada.

Este tipo de editor se denomina "full screen" o de *pantalla completa,* debido a que la edición puede realizarse en cualquier punto de la pantalla.

Los editores de pantalla presentan algunas limitaciones en ciertos casos. Por ejemplo, hay equipos que permiten introducir caracteres de control dentro del texto asociado a una instrucción PRINT; caracteres que se obtienen directamente mediante el uso de las teclas del cursor y otras especiales. De esta forma, cuando se está escribiendo una constante alfanumérica (ello viene señalado por las comillas que preceden al texto escrito), las teclas del cursor no realizarán la acción prevista, sino que escribirán el carácter de control correspondiente. Ello impedirá el libre movimiento del cursor en dicha zona.

Otros ordenadores incorporan un segundo tipo de editor: el denominado *editor de líneas,* cuya actividad está regida por el comando EDIT. En este caso, sólo es posible editar la línea seleccionada por medio del comando EDIT. La selección se realiza colocando el número de la línea a editar tras el referido comando. A continuación, pueden ya realizarse las correcciones oportunas, apoyándose en el repertorio de subcomandos del editor.

EL COMANDO EDIT

EDIT es el comando básico entorno al que se organiza el funcionamiento del *editor de líneas.* Al ejecutarlo, se accede al denominado *modo de edición:* situación que permite operar las modificaciones necesarias en la línea de programa seleccionada.

El formato de una instrucción del tipo EDIT es el siguiente:

EDIT <número de línea>

El número de línea debe coincidir con el de la línea de programa a editar.

Tras la ejecución de este comando, la línea seleccionada entra en modo edición.

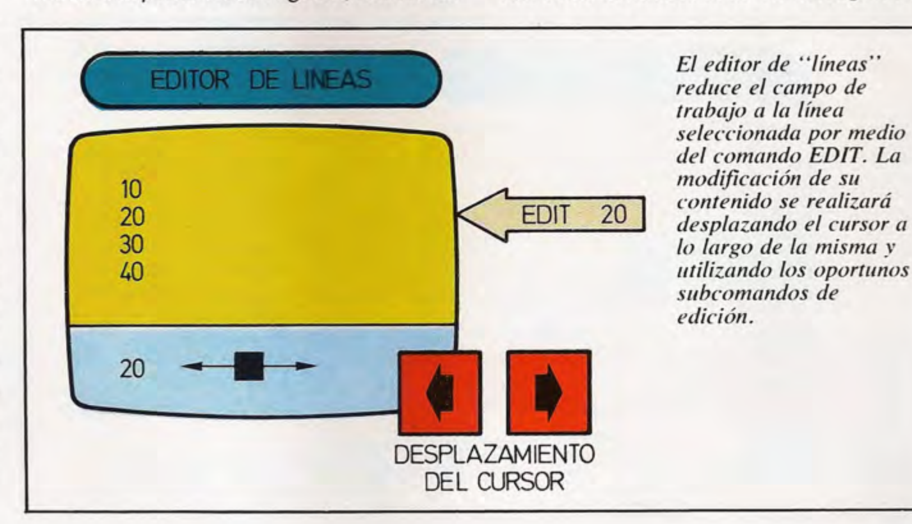
Algunos editores de líneas muestran la línea indicada en la zona inferior de la pantalla, lista para su edición. En otros, únicamente aparecerá el número de la línea de edición y el cursor situado inmediatamente después. Acto seguido, será necesario seleccionar una de las diversas opciones de edición disponibles.

El editor de líneas exige teclear el comando EDIT cada vez que se desee editar una nueva línea. No obstante, una vez que se ha entrado en modo edición se permite el libre movimiento del cursor a través de la línea seleccionada. En esta situación, algunas teclas específicas dan paso a las funciones de inserción, borrado o sustitución de caracteres.

EL EDITOR DE "BUFFER"

Existe un tercer tipo de editor, tal vez el más primitivo de los mencionados.

Para manejarlo es preciso situarse al principio de la línea a editar (en cualquier punto de la pantalla), e ir "copiando" los caracteres en un "buffer" o memoria temporal. Esto se consigue "pasando por encima" de los mencionados caracteres con el cursor. A medida que se van dando las órdenes adecuadas se consigue la in-



El editor de "líneas" reduce el campo de trabajo a la línea seleccionada por medio del comando EDIT. La modificación de su contenido se realizará desplazando el cursor a lo largo de la misma y utilizando los oportunos subcomandos de edición.

serción o el borrado de los caracteres necesarios.
El principal inconveniente de los editores que utilizan buffer es la necesidad de repasar toda la línea. Esta característica obliga a llegar al final de la misma antes de poder accionar la tecla RETURN.
Semejante imperativo deriva de la forma en la que se aceptan los caracteres en edición. El editor dispone al efecto de una zona de memoria denominada "buffer", de la cual se extrae la línea editada al concluir el proceso de edición accionando la tecla RETURN.
Para introducir caracteres en el buffer es necesario pasar el cursor sobre ellos. Si se teclea algún carácter nuevo, éste entrará en el buffer inmediatamente. La inserción o sustitución de caracteres se habilita por medio de las teclas de desplazamiento del cursor, posicionando a éste en el lugar adecuado. Ello permite saltar porciones de la línea que no serán introducidas en el buffer. No cabe duda que este método de edición exige una profunda concentración por parte del programador; éste ha de llevar en mente los caracteres introducidos en el buffer.

LA DIVERSIDAD DE LOS EDITORES

Son varios los tipos de editores de programas que cabe encontrar en distintos ordenadores. Salvando las peculiaridades que pueden presentar editores de un mismo tipo, cabe considerar los tres grupos básicos definidos hasta el momento:

—Editores de pantalla (totales o "full screen").
—Editores de líneas.
—Editores de "buffer" o memoria temporal.

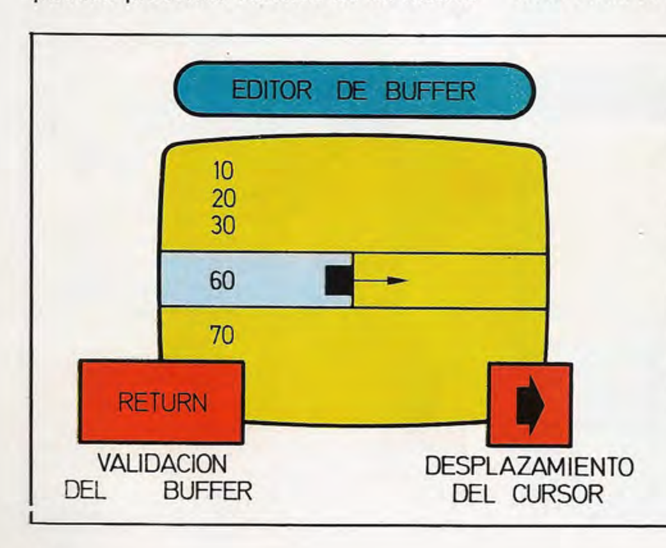
En general, el editor de uso más fácil es el de pantalla, puesto que permite una edición inmediata. Permite corregir cualquier instrucción presente en pantalla sin más que accionar la tecla de retroceso de carro tras realizar las correcciones oportunas.
Los editores que utilizan un buffer son los más incómodos y cuyo empleo es menos inmediato; no obstante, una vez que su manejo resulta familiar constituirán también una eficaz herramienta de trabajo.
Los editores de línea son, en principio, los menos potentes; sin embargo, de ellos se puede conseguir mucho más de lo que parece a primera vista. Existen muchas variantes, aunque, por lo general, hay que entrar en el modo de edición por medio de una orden específica (normalmente EDIT).
Al margen del tipo de editor que incorpore cada dialecto BASIC, existen ciertos comandos de apoyo a las tareas de edición de programas que pueden estar presentes en distintos equipos. Este grupo de comandos resultan adecuados para numerar automáticamente las sucesivas líneas de un programa en edición, para su renumeración automática, o para el borrado directo de líneas de programa.

EL COMANDO AUTO

A menudo resulta tedioso el hecho de tener que teclear el número de línea cuando se introduce un programa por vez primera. Incomodidad que aumenta a medida que el programa es más extenso. Para facilitar esta tarea, algunos traductores BASIC disponen del comando AUTO.
El referido comando libera al usuario de la necesidad de llevar la cuenta de los números de línea, puesto que se encarga de numerar las líneas a medida que se introducen, calculando el número correspondiente a cada línea sucesiva.
El formato de una instrucción construida a partir del comando AUTO es el siguiente:

AUTO [<número de línea>[,<incremento>]]

Una vez ejecutado en modo directo, el ordenador generará de forma automática el número de línea correspondiente tras cada acción del programador sobre la tecla de retroceso de carro (CR).
AUTO comienza la numeración del programa a partir del número indicado en el argumento: <número de línea>. Este valor crecerá en las sucesivas líneas en el número de unidades que se hayan indicado en la zona de <incremento>.



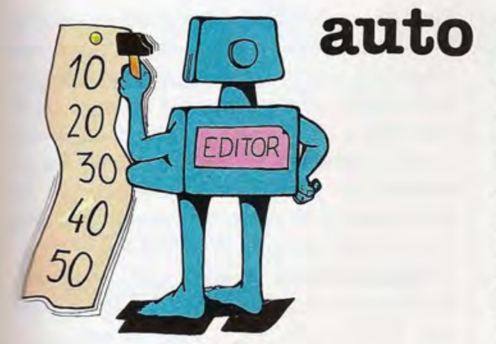
El editor menos evolucionado y más incómodo para el usuario es el de "buffer". Los caracteres de la línea en edición barridos por el cursor, y las modificaciones introducidas, se copian en una memoria temporal. El contenido de esta última será aceptado como definitivo al accionar la tecla RETURN o ENTER.

AUTO

Activa el modo de numeración automática de líneas.

Formato: AUTO [*<número de línea>*[,*<incremento>*]]

Ejemplos: AUTO
AUTO 10
AUTO 50,10



Cuando la instrucción AUTO se formula desprovista de argumento, el ordenador considera que ambos valores (<número de línea> e <incremento>) adoptan el valor 10. Por ello, numerará el programa a partir de la línea 10 y en sucesivos incrementos de 10 unidades.

Si el <número de línea> va seguido por una coma y no se especifica el incremento, se entiende que el valor del incremento es igual al incremento indicado en el último comando AUTO ejecutado con anterioridad.

EL COMANDO RENUM

A la hora de intentar pulir y perfeccionar un determinado programa BASIC, es fundamental entender la función de cada zona del mismo, así como las relaciones entre ellas. Para ello no basta con conocer a fondo el lenguaje. Muchas veces, la complejidad de su estructura no permite un seguimiento fácil de la ejecución.

La inserción de comentarios (REM) suele aclarar el cometido de los distintos fragmentos del programa. Otro método clarificador consiste en agrupar las líneas del programa en bloques fraccionados. Por ejemplo, una cierta rutina o zona de cálculo específica se puede situar en las líneas 2000 y siguientes, otra a partir de la 3000, etc. Posteriormente, será posible identificar las distintas partes del programa atendiendo exclusivamente a las primeras cifras de los números de línea.

Habitualmente, las líneas intercaladas contribuyen al desorden general, alterando la pauta de numeración. En determinados casos, incluso se hace imposible insertar una nueva línea: cuando los números de dos líneas consecutivas difieren en una sola unidad.

Ambas situaciones hacen aconsejable —cuando no obligan— a "renumerar" las líneas del programa; esto es: a cambiar los números de las líneas sin alterar el orden de las mismas.

El comando utilizable a tal efecto es RENUM. Este permite sustituir los números de línea contiguos por otros nuevos, especificando, además, el salto deseado entre cada dos números consecutivos. Su formato general es:

RENUM [[<nuevo>][,[<anterior>]-[,<incremento>]]]

Como se ha indicado, su misión es la de sustituir los números de línea situados a partir del indicado en la zona <anterior>, hasta el final del programa, por los números <nuevo> y sucesivos, en saltos de tantas unidades como especifique la zona de <incremento>.

El número indicado en el campo <anterior> hace referencia al primer número de línea a sustituir. Semejante precisión permite renumerar sólo una parte del programa, manteniendo en otras zonas la numeración de líneas original.

El siguiente ejemplo ilustra la actuación de este tipo de instrucciones. El objeto de trabajo es un simple programa capaz de calcular el cuadrado del número que se introduzca como respuesta a la instrucción INPUT. El programa finalizará al recibir un cero como dato de entrada.

```
 10 REM COMANDO RENUM
 20 PRINT "PROGRAMA EJEMPLO"
 21 INPUT A
 22 IF A=0 THEN GOTO 158
 23 LET B=A*A
122 PRINT "EL CUADRADO DE";A
157 PRINT "ES:";B
158 END
■
```

Subcomandos del editor de líneas

Los editores de líneas suelen disponer de un conjunto de subcomandos que facilitan la edición de la línea en curso. Estos subcomandos se detallan a continuación, agrupados de acuerdo a la función que realizan:

CURSOR

Espacio. Mueve el cursor un lugar hacia la derecha, mostrando el carácter que ocupa dicha posición en la línea original.

INSERCION

I. Permite insertar tantos caracteres como se desee a partir de la posición en la que esté situado el cursor. Para salir de esta opción es necesario pulsar la tecla ESCAPE.

X. Extensión de línea. Desplaza el cursor al final de la línea y entra en inserción en ese punto.

BORRADO

D: Borra el carácter situado en la posición siguiente a la que ocupa el cursor.

H: Borra todos los caracteres situados a la derecha del cursor y entra automáticamente en modo inserción.

S: Va acompañado de un número y un carácter. Buscará el carácter indicado tantas veces como señale el número introducido y colocará el cursor delante del carácter afectado.

K: Actúa de forma análoga al subcomando S, si bien, borra todos los caracteres por los que pasa.

SUSTITUCION

C: Sustituye el carácter situado a la derecha del cursor por el que se introduzca a continuación de C; puede acompañarse por un parámetro que indicará el número de caracteres contiguos a sustituir.

FINAL

(CR): Señala el final del proceso de edición; al accionar la tecla en cuestión (RETURN o ENTER) el ordenador aceptará todos los cambios realizados.

E: Salida de las modificaciones introducidas.

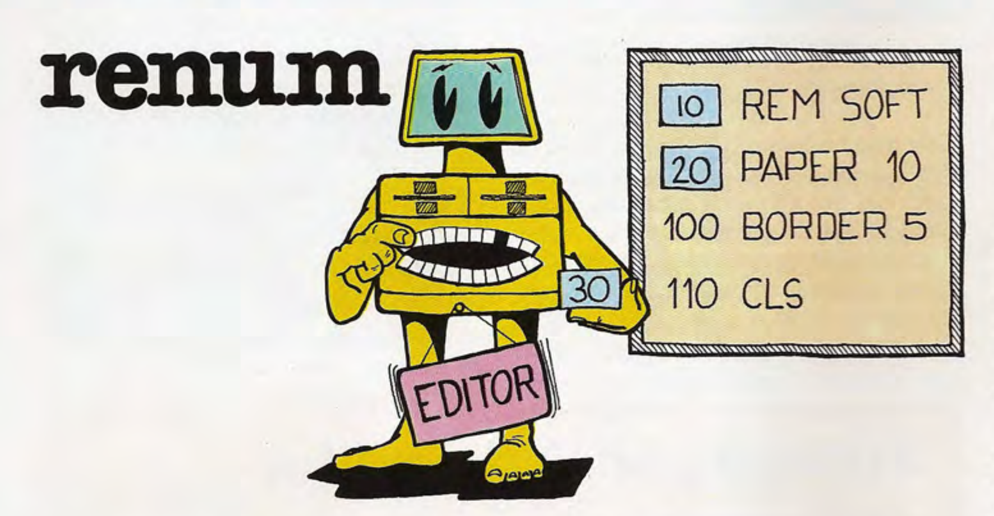
L: Lista la línea en curso y entra en edición la siguiente.

RENUM

Renumera líneas de programa.

Formato: RENUM [[<*número nuevo*>] [,[<*número antiguo*>] [,<*incremento*>]]]

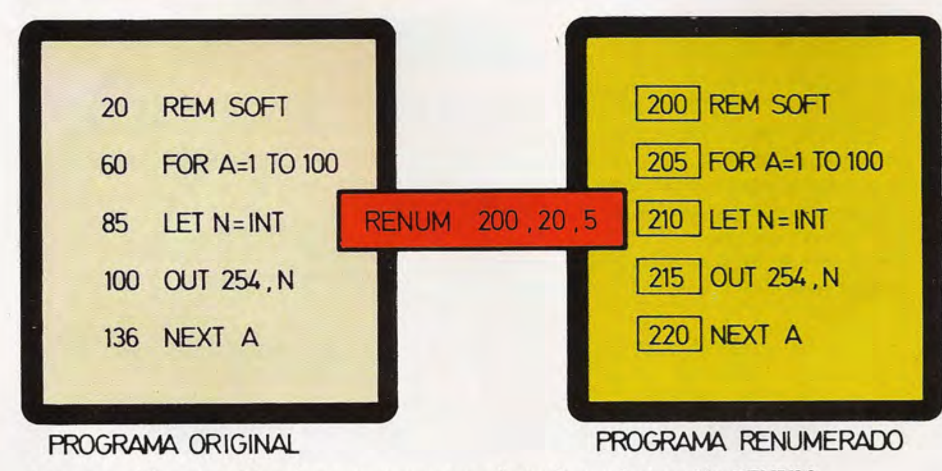
Ejemplos: RENUM 100
RENUM 100,10
RENUM 100,10,5



Un programa de indudable sencillez, pero que ganaría en claridad con la inserción de algunos comentarios.
El problema surge en la peculiar numeración de sus líneas. ¿Cómo es posible emplazar una instrucción entre las líneas 20 y 21, o entre la 21 y 22? No queda más alternativa que abrir espacio entre las mencionadas líneas; una solución que el comando RENUM pondrá en práctica de forma inmediata y automática:

```
RENUM 100,21,2

LIST
 10 REM COMANDO RENUM
 20 PRINT "PROGRAMA EJEMPLO"
100 INPUT A
102 IF A=0 THEN GOTO 110
104 LET B=A*A
106 PRINT "EL CUADRADO DE";A
108 PRINT "ES:";B
110 END
```



Renumeración automática de un programa por efecto de una instrucción RENUM.

La instrucción RENUM formulada, ordena la remuneración de la zona de programa situada a partir de la línea 21. La nueva numeración debe partir del número de línea 100 y distribuirse en sucesivas líneas cuyos números se distancien en incrementos de 2 unidades. La ejecución de una orden LIST permite observar la nueva distribución de números de línea.
Ahora es posible ya emplazar nuevas instrucciones precediendo a las instrucciones INPUT e IF/THEN. Por ejemplo:

```
 95 PRINT "INTRODUZCA UN NUMERO"
101 REM DETECCION DE FIN DE PROGRAMA

LIST
 10 REM COMANDO RENUM
 20 PRINT "PROGRAMA EJEMPLO"
 95 PRINT "INTRODUZCA UN NUMERO"
100 INPUT A
101 REM DETECCION DE FIN DE PROGRAMA
102 IF A=0 THEN GOTO 110
104 LET B=A*A
106 PRINT "EL CUADRADO DE";A
108 PRINT "ES:";B
110 END
```

Un nuevo uso de RENUM permitirá remodelar la numeración de líneas logrando una distribución consecutiva en saltos de diez unidades:

```
RENUM 30,95,10

LIST
 10 REM COMANDO RENUM
 20 PRINT "PROGRAMA EJEMPLO"
 30 PRINT "INTRODUZCA UN NUMERO"
 40 INPUT A
 50 REM DETECCION DE FIN DE PROGRAMA
 60 IF A=0 THEN GOTO 100
 70 LET B=A*A
 80 PRINT "EL CUADRADO DE";A
 90 PRINT "ES:";B
100 END
```

El programa está ya completo y organizado con la colaboración del comando RENUM. Un comando cuya actuación va más allá de la simple modificación de los números que preceden a las líneas de programa. Como puede observarse a lo largo de las sucesivas operaciones de renumeración, el propio RENUM se ha encargado de actualizar el número de línea al que debe realizarse el salto ordenado en la instrucción IF/THEN (línea 22 del listado original). En cada caso, el número de línea asociado a la zona GOTO ha adoptado la referencia de la instrucción END.

TABLA DE CONVERSION

ORDENADOR	EDIT	AUTO	RENUM	DELETE
	EDIT <nl>	**AUTO <nl>, <inc.>**	**RENUM <nue.>, <ant.>, <inc.>**	**DELETE <inic.>–<fin.>**
APPLE II (APPLESOFT)	—	—	—	DELETE <inic.>–<fin.>
APRICOT (M-BASIC)	EDIT <nl>	AUTO <nl>, <inc.>	RENUM <nue.>, <ant.>, <inc.>	—
ATARI	—	—	—	—
CBM 64	—	—	—	DEL <inic.>–<fin.>
DRAGON	EDIT <nl>	—	RENUM <nue.>, <ant.>, <inc.>	DELETE <inic.>–<fin.>
EQUIPOS MSX	—	AUTO <nl>, <inc.>	RENUM <nue.>, <ant.>, <inc.>	DELETE <inic.>–<fin.>
HP-150	EDIT <nl>	AUTO <nl>, <inc.>	RENUM <nue.>, <ant.>, <inc.>	DELETE <inic.>–<fin.>
IBM PC	EDIT <nl>	AUTO <nl>, <inc.>	RENUM <nue.>, <ant.>, <inc.>	DELETE <inic.>–<fin.>
MPF	—	—	—	DEL <inic.>–<fin.>
NCR DM-V (MS-BASIC)	EDIT <nl>	AUTO <nl>, <inc.>	RENUM <nue.>, <ant.>, <inc.>	DELETE <inic.>–<fin.>
NEW BRAIN	—	—	—	DELETE <inic.>–<fin.>
ORIC	EDIT <nl>	AUTO <nl>, <inc.>	—	—
SHARP MZ-700 (MZ-BASIC)	—	AUTO <nl>, <inc.>	RENUM <nue.>, <ant.>, <inc.>	DELETE <inic.>–<fin.>
SINCLAIR QL	EDIT <nl>, <inc.> (1)	AUTO <nl>, <inc.>	RENUM <inic.>TO<fin.>; <nue.>, <inc.> (1)	DLINE <inic.>TO<fin.> (1)
SPECTRAVIDEO	—	AUTO <nl>, <inc.>	RENUM <nue.>, <ant.>, <inc.>	DELETE <inic.>–<fin.>
ZX-SPECTRUM	EDIT (2)	—	—	—

<nl>: Número de línea. <inc.>: Incremento. <nue.>: Nuevo número de línea. <ant.>: Número de línea antiguo. <inic.>: Número de línea inicial. <fin.>: Número de línea final.

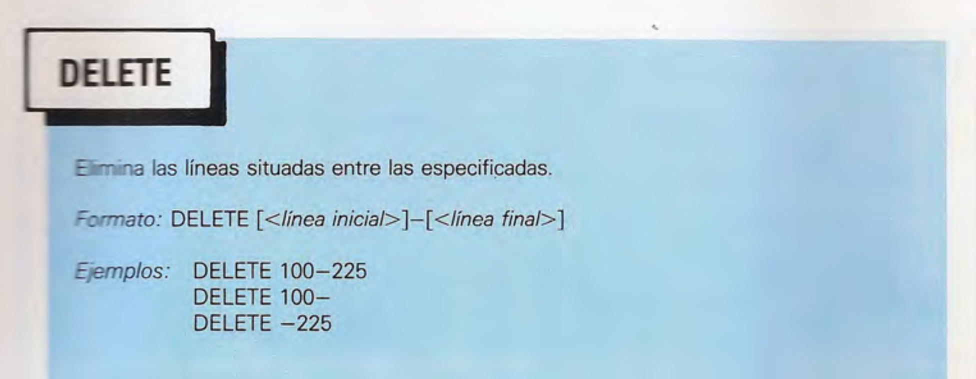
FORMULACIONES DE LOS COMANDOS

EDIT <NL>: Edita la línea cuyo número se indica. AUTO <nl>, <inc.>: Genera automáticamente los sucesivos números de línea tras cada pulsación de la tecla RETURN. La numeración empieza a partir de <nl>, en sucesivos incrementos de <inc.> unidades. RENUM <nue.>, <ant.>, <inc.>: Renumera las líneas del programa a partir de la indicada en <ant.>, situándolas a partir de la línea <nue.> en incrementos de <inc.> unidades. DELETE <inic.>–<fin.>: Borra el bloque de líneas comprendidas entre las dos especificadas.

OBSERVACIONES

(1) SINCLAIR QL
EDIT <nl> [, <inc.>]: Edita la línea <nl> y sucesivas en incrementos de <inc.>. RENUM [<inic.> [TO <fin>;]][<nue>][, <inc.>]: Renumera automáticamente desde la línea <inic.> hasta la <fin.>, tomando como base de la nueva numeración la línea <nue.> en incrementos de <inc.> unidades. DLINE <inic.> TO <fin.>: Borra el bloque de líneas indicado. Pueden encadenarse expresiones de este tipo, separando por comas los rangos de líneas a borrar.

(2) ZX-SPECTRUM
El comando EDIT no lleva asociado un número de línea. No obstante, hay que situar un cursor que aparece en pantalla sobre la línea a editar.

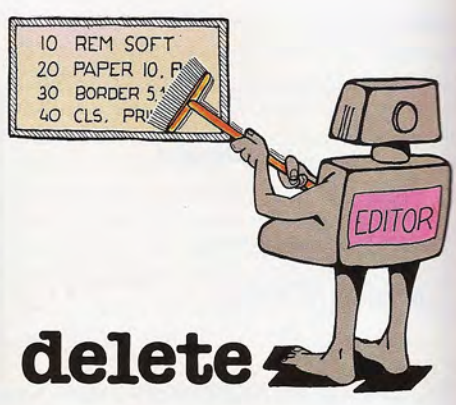


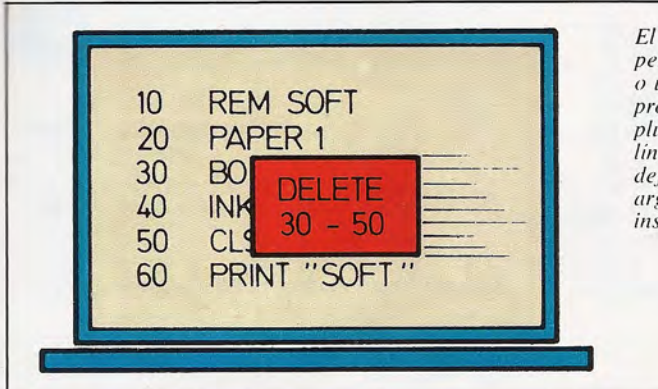
DELETE

Elimina las líneas situadas entre las especificadas.

Formato: DELETE [<*línea inicial*>]–[<*línea final*>]

Ejemplos: DELETE 100–225
DELETE 100–
DELETE –225





El comando DELETE permite borrar una línea o un grupo de líneas del programa "de un plumazo". El bloque de líneas a eliminar debe definirse en el argumento de la referida instrucción.

BORRADO DE LINEAS

A la hora de depurar un programa, tan importante es la posibilidad de intercalar nuevas líneas como la de borrar las antiguas. Esta segunda alternativa se consigue por el simple procedimiento de teclear el número correspondiente a la línea en cuestión, seguido por una acción sobre la tecla RETURN. Ello equivale a sustituir la línea especificada por una línea vacía de contenido.
Este método resulta útil y rápido cuando la cantidad de líneas a borrar no es muy grande. Sin embargo, cuando se trata de borrar amplias zonas del programa, el trabajo puede llegar a resultar agotador. Para este cometido existe un comando BASIC experto en el borrado de varias líneas a la vez: DELETE. Su formato como instrucción directa es el siguiente:

DELETE <inicial>–<final>

Los campos que siguen al comando determinan las líneas que deben desaparecer del programa. El primer número especifica el número de línea *inicial* o inferior del bloque a eliminar, y el segundo el número de línea *final* o mayor de los afectados por el borrado.
Partiendo del último listado remodelado por medio del comando RENUM, puede observarse el efecto de la omisión de uno de los campos en la instrucción DELETE.

```
DELETE 30

LIS
 10 REM COMANDO RENUM
 20 PRINT "PROGRAMA EJEMPLO"
 40 INPUT A
 50 REM DETECCION DE FIN DE PROGRAMA
 60 IF A=0 THEN GOTO 100
 70 LET B=A*A
 80 PRINT "EL CUADRADO DE";A
 90 PRINT "ES:";B
100 END
```

Una de las posibles formulaciones parciales de la instrucción DELETE es la que incorpora un sólo número de línea en el argumento. Como revela el ejemplo anterior, su efecto se traduce exclusivamente en el borrado de la línea indicada.
Otra de las posibles variantes de DELETE es la que muestra el siguiente ejemplo (DELETE 80–). La ausencia de número de línea final supone el borrado de todas las líneas del programa localizadas a partir de la inicial (línea 80 en el ejemplo):

```
DELETE 80–

LIST
10 REM COMANDO RENUM
20 PRINT "PROGRAMA EJEMPLO"
40 INPUT A
50 REM DETECCION DE FIN DE PROGRAMA
60 IF A=0 THEN GOTO 100
70 LET B=A*A
```

Una nueva ejecución de la instrucción DELETE, esta vez desprovista de número de línea inicial (DELETE –20), conduce al borrado de las líneas que van desde el principio del programa hasta la indicada (línea 20):

```
DELETE –20

LIST
40 INPUT A
50 REM DETECCION DE FIN DE PROGRAMA
60 IF A=0 THEN GOTO 100
70 LET B=A*A
```

Por último, ya sólo queda por observar la actuación de DELETE con su formato de instrucción más genérico:

```
DELETE 50–60

LIST
40 INPUT A
70 LET B=A*A
■
```

Logo (5)

Los procedimientos del LOGO

A la hora de definir las características primordiales del LOGO, surge de inmediato el calificativo de lenguaje *modular*. En efecto, los programas en LOGO están construidos a base de módulos cuya conjunción da lugar a estructuras más complejas. Estos módulos son los denominados *procedimientos;* una de las facetas más importantes del LOGO, cuya definición y empleo va a ser objeto de estudio en el presente capítulo.

CREACION DE PROCEDIMIENTOS

Un *procedimiento* no es más que una sucesión de órdenes encaminadas a realizar una acción más compleja. Requisito básico es que el conjunto de órdenes que constituyen el procedimiento aparezcan ordenadas en una secuencia apropiada; hay que tener en cuenta que no es lo mismo sumar dos cantidades y elevar el resultado al cubo, que elevar al cubo las cantidades y luego sumar los resultados.
Las órdenes de un procedimiento, una vez ejecutadas, darán lugar a un resultado; éste coincidirá con la acción encomendada al procedimiento. Tal acción puede ser, sencillamente, una parte de un cálculo más complejo. Por ejemplo, un procedimiento que resuelva ecuaciones de segundo grado resultará de gran utilidad a la hora de hallar solución a múltiples problemas matemáticos.
Todo procedimiento LOGO comienza con la palabra TO y finaliza con END. La primera línea debe contener, tras la palabra clave TO, el nombre con el que se va a identificar al procedimiento. Ello indica al LOGO que a partir de ese punto empieza el procedimiento especificado.
Las restantes líneas estarán ocupadas por las sucesivas órdenes, con una condición importante: la primera orden de cada línea debe ser un *comando*. En la última línea se colocará la palabra clave END, para que el LOGO entienda que ha concluido la definición del procedimiento.

```
TO HOLA
PRINT "HOLA
PRINT [BIENVENIDO AL LOGO]
END

HOLA DEFINED
```

El programa corresponde a la creación de un procedimiento. En efecto, cabe observar que se han respetado las condiciones impuestas para la definición. La primera línea empieza con la palabra TO, seguida por el nombre del procedimiento —HOLA, en nuestro caso—, y la última línea incluye la orden END; esta última comunica al ordenador que ha terminado la definición.
La respuesta de la máquina no se hace esperar: muestra en la pantalla el mensaje "HOLA DEFINED" (HOLA definido).
A partir de este preciso instante, para hacer uso del procedimiento HOLA, bastará con teclear su nombre.
Realmente, la creación de un procedimiento equivale a definir una nueva orden en el vocabulario LOGO; una orden que funcionará exactamente igual que las incluidas en el repertorio original del LOGO. De hecho, un procedimiento puede incluso formar parte de otro procedimiento más complejo (superprocedimiento). Veamos un nuevo ejemplo:

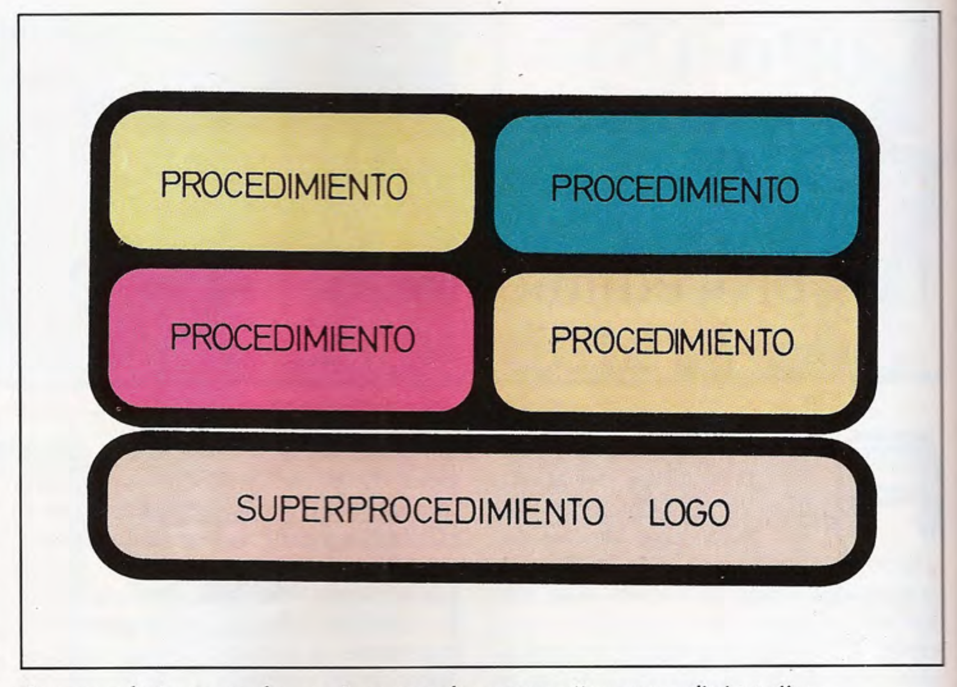


Los programas LOGO son estructuras creadas a partir de la asociación de procedimientos o módulos elementales. Un procedimiento es un conjunto de órdenes destinadas a programar una determinada acción.

```
TO SALUDO
HOLA
PRINT [COMO TE LLAMAS?]
PRINT LPUT FIRST RL [BUENOS DIAS]
END

SALUDO DEFINED
```

En esta ocasión se ha definido un nuevo procedimiento, denominado SALUDO. Un dato significativo es que se ha utilizado el procedimiento anterior, HOLA, como si se tratara de un *comando* propio del LOGO. Esta filosofía permite la subdivisión de problemas complicados en pequeños fragmentos elementales.
Por ejemplo, para calcular los gastos domésticos cabe seguir los siguientes pasos:



Los procedimientos pueden asociarse para dar cuerpo a "superprocedimientos" o procedimientos LOGO más complejos.

— Apuntar gastos de alimentación.
— Sumarlos y apuntar el resultado.
— Apuntar gastos de servicios contratados para la casa (luz, agua...).
— Sumar y escribir resultado.
— Apuntar gastos de otras compras.
— Sumar y apuntar resultado.
— Sumar los totales y escribir el resultado.



La modularidad del lenguaje LOGO queda plasmada en los procedimientos. Estos son módulos que pueden integrarse sucesivamente en otros procedimientos, cada vez más complejos, hasta constituir el programa.

La tarea aparece fraccionada en un conjunto de acciones elementales que, sin lugar a dudas, facilitan la comprensión y el cálculo. Esta misma filosofía de descomponer un trabajo en acciones parciales es compartida por la programación en LOGO. El caso propuesto puede adoptar la forma de procedimiento LOGO:

```
TO GASTOS
ENTRA.ALIMEN
SUMA.ALIMEN
ENTRA.CASA
SUMA.CASA
ENTRA.OTROS
SUMA.OTROS
SUMA.TOTAL
END
```

El procedimiento global GASTOS incluye las sucesivas acciones a realizar para evaluar el gasto doméstico total; desde luego, hace uso de otros procedimientos más elementales que será preciso definir por separado. En el ejemplo, se observa que todas las órdenes de GASTOS son procedimientos. Todas ellas deben ser creadas antes de utilizar el procedimiento GASTOS.

La definición del último de ellos, SUMA. TOTAL, puede adoptar la siguiente forma:

```
TO SUMA.TOTAL
MAKE "TOTAL SUM :OTROS :CASA
:ALIMENTOS
PRINT [TOTAL GASTOS]
PRINT LIST :TOTAL "PTS
END

SUMA.TOTAL DEFINED
```

Las variables OTROS, CASA y ALIMENTOS, utilizadas para calcular la suma total, contienen las sumas parciales de los respectivos conceptos. Por ejemplo, la variable :CASA que almacena el gasto total de los servicios domésticos contratados se calculará dentro del procedimiento SUMA.CASA:

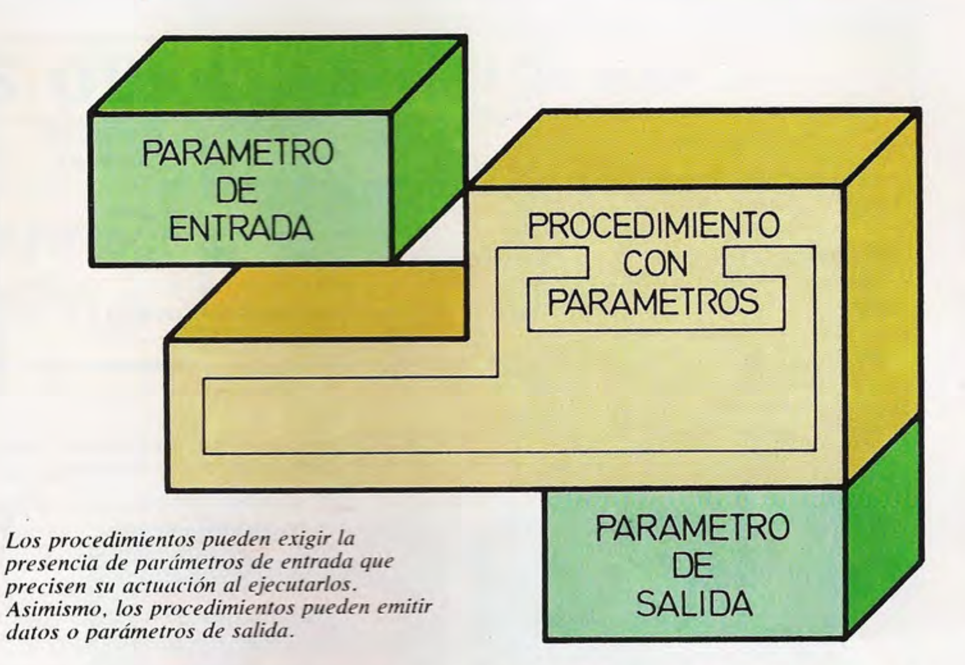
```
TO SUMA.CASA
MAKE "CASA SUMLIST :LISTA.CASA
PRINT [TOTAL GASTOS DE CASA]
PRINT LIST :CASA "PTS
END
```

El procedimiento SUMA.CASA hace uso del subprocedimiento SUMLIST a modo de operador. SUMLIST admite una entrada y proporciona una salida; este tipo de procedimientos, que pueden actuar indistintamente como comandos u operadores, reciben el nombre de *procedimientos con parámetros.*

PROCEDIMIENTOS CON PARAMETROS

Al igual que ocurre, en el caso general, con los datos del LOGO, cabe distinguir dos tipos de parámetros: de entrada y de salida. Los primeros constituyen datos de entrada al procedimiento y deben especificarse detrás del propio nombre del procedimiento. Por ejemplo:
TO SUMLIST :L
indica que el procedimiento SUMLIST precisa de un dato de entrada representado por la variable :L. En el mismo caso, el dato en cuestión será identificado, dentro del procedimiento, por :L.
El procedimiento SUMLIST podría definirse de la siguiente forma:

```
TO SUMLIST :L
IF EMPTYP :L [OUTPUT 0 STOP]
OUTPUT SUM (FIRST :L) (SUMLIST
(BUTFIRST :L))
END
```



Los procedimientos pueden exigir la presencia de parámetros de entrada que precisen su actuación al ejecutarlos. Asimismo, los procedimientos pueden emitir datos o parámetros de salida.

Este procedimiento es algo más complicado; si bien, por el momento sólo se prestará atención a los parámetros de entrada y salida.
El parámetro de entrada se ha utilizado como dato en los cálculos. Su empleo es idéntico al de las variables; ello permite que el procedimiento admita distintos datos como entrada.
Respecto a la salida, hay que mencionar al operador OUTPUT (salida). OUTPUT coloca su dato de entrada como dato de salida del procedimiento en el que se encuentra. En el ejemplo propuesto, el mencionado operador se utiliza para "extraer" el valor de la suma. Por supuesto, un determinado procedimiento puede carecer de uno o de ambos tipos de parámetros.
A continuación aparecen dos ejemplos prácticos que ilustran la actuación del operador OUTPUT y resumen el comportamiento de un procedimiento con parámetros.
El primero de ellos adjudica el dato numérico 3 al procedimiento TRES; una operación análoga a las asignaciones de variables, tal como se observa al ordenar la impresión de TRES:

```
TO TRES
OUTPUT 3
END
TRES DEFINED

PRINT TRES
3
```

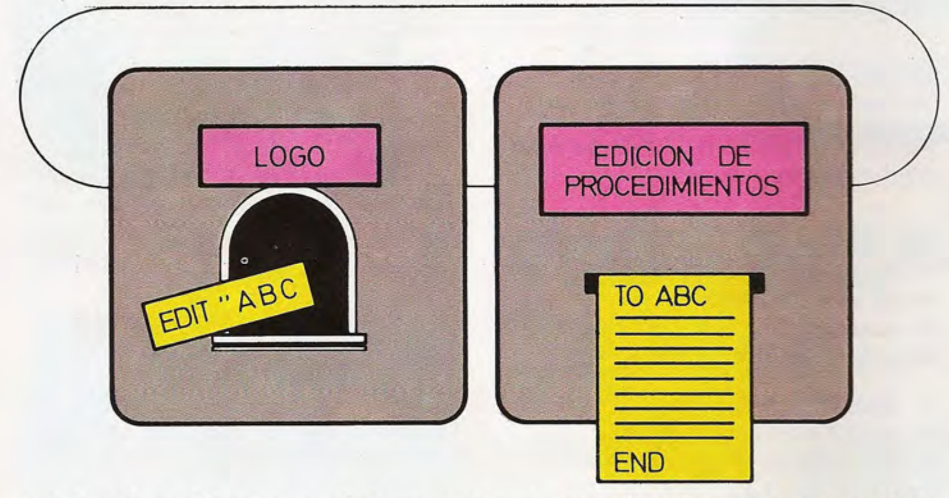
El segundo ejemplo, algo más evolucionado, ilustra la actuación de un procedimiento con parámetros. En primer lugar, se procede a la definición del procedimiento INICIAL, asociándole un parámetro: PALABRA y se define la función de imprimir el primer elemento del citado parámetro.

```
TO INICIAL :PALABRA
PRINT FIRST :PALABRA
END

INICIAL DEFINED
```

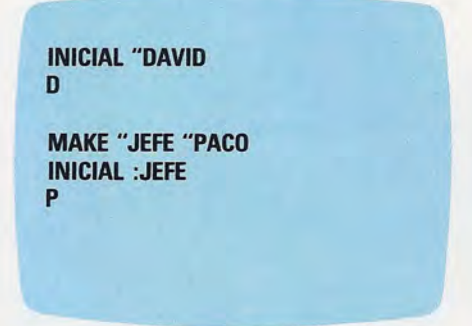
TABLA DE ORDENES-LOGO		
INSTRUCCION	COMETIDO	OPERADOR/ COMANDO
MAKE <palabra> <dato>	Crea variable	Comando
TO <nombre> [<variables>]	Crea procedimiento	Comando
END	Determina el fin de un procedimiento	Comando
EDIT [<nombre>...]	Activa el editor de procedimientos	Comando

<nombre>: nombre del procedimiento (sin comillas).
<variables>: posibles variables locales (precedidas por dos puntos).



A la hora de modificar un procedimiento, no es preciso reescribirlo de nuevo en su integridad. El lenguaje LOGO dispone de un comando de edición (EDIT) que visualiza el contenido del procedimiento elegido y permite su modificación.

Una vez definido el procedimiento podemos ya utilizarlo en la práctica, sin más que recurrir a su nombre como si se tratara de una orden LOGO.



En el primer caso, se le otorga como dato de entrada la palabra DAVID. Su ejecución mostrará en pantalla el primer elemento (letra D) del dato entregado como parámetro.
La entrada asociada al procedimiento con parámetro INICIAL puede ser también una variable. La asignación MAKE, otorga la palabra PACO a la variable JEFE. Ahora, al ejecutar el procedimiento INICIAL sobre el parámetro :JEFE, la pantalla mostrará la primera letra del nombre PACO contenido en la variable JEFE.

EDICION DE PROCEDIMIENTOS

Si se desea modificar un procedimiento no hace falta escribirlo de nuevo. El comando EDIT facilita esta tarea, visualizando y permitiendo la modificación de procedimientos. EDIT admite un dato de entrada que puede coincidir con un nombre de procedimiento (palabra) o con varios nombres (lista).
Realmente, EDIT abre el acceso al *editor de procedimientos.* Una vez en el editor, se visualiza el procedimiento o procedimientos especificados y el cursor. Este último puede posicionarse allí donde el usuario desee realizar el cambio.
Es posible editar más de un procedimiento a la vez. Para ello es preciso confeccionar una lista con los nombres de los procedimientos y situarla como dato de entrada al comando EDIT. Por ejemplo:

```
EDIT "TRES
EDIT [GASTOS SUMA.TOTAL SUMA.CASA SUMLIST]
```

son formulaciones correctas del mencionado comando. En el primer caso facilitará la modificación del procedimiento cuyo nombre es TRES, mientras que en el segundo permitirá la edición de los cuatro procedimientos cuyos nombres figuran en la lista.



La orden EDIT vuelca el procedimiento en la pantalla y brinda al usuario la posibilidad de corregir y alterar su contenido. La edición se reduce a desplazar el cursor al punto adecuado y utilizar las teclas alfabéticas y numéricas para realizar los cambios oportunos.

Los ficheros del CP/M

Manejo de la información en la memoria externa

El sistema operativo CP/M fracciona las tareas cuya responsabilidad tiene encomendada en tres grandes bloques. Estos guardan una perfecta consonancia con la estructura modular del CP/M, de tal forma que cada bloque de tareas es gestionado por uno de los tres módulos constitutivos: CCP (procesador de comandos de consola), BDOS (sistema operativo básico de disco) y BIOS (sistema básico de entrada/salida).

GESTION DE INFORMACION EN LA MEMORIA EXTERNA

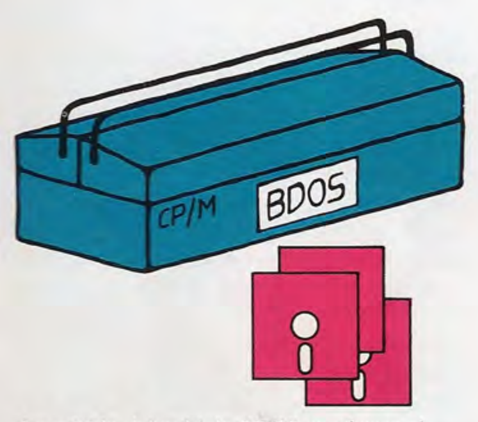
La misión del BDOS es ofrecer al usuario los medios adecuados para un manejo simple y eficaz de la información almacenada en los discos del sistema. Para ello, brinda un determinado número de funciones, previamente programadas en el sistema operativo, que permiten la manipulación de ficheros y registros, y liberan al usuario de la preocupación de generar el espacio requerido para el almacenamiento de ficheros en el disco.

A diferencia con el CCP, el módulo BDOS no está abierto a un acceso directo a base de comandos introducidos a través del teclado; su acceso es más arduo, puesto que hay que descender hasta el nivel del lenguaje ensamblador.

En todo caso, el propio sistema operativo apoya esta actividad. La ejecución de las funciones del BDOS se ve facilitada por la existencia de ciertas rutinas dentro del sistema operativo. Estas trasladan a un registro de almacenamiento interno del microprocesador un indicativo de la función que se desea realizar y apelan a la actividad del BDOS; de inmediato, éste ejecutará la tarea precisada por el código de función y, acto seguido, devolverá el control al programa desde el cual fue llamado.

Los ficheros que crea y con los que opera el sistema operativo CP/M, están constituidos por una colección de *registros* de 128 bytes cada uno. Cada fichero puede incorporar un máximo de 65.536 registros; ello significa que el tamaño máximo de un fichero CP/M es de 128 × 65.536 = 8.378.508 bytes, o lo que es lo mismo 8 Megabytes.

Las operaciones de lectura y escritura no se efectúan adoptando como patrón el registro, sino que se realizan a un nivel de los denominados *bloques*. Un bloque equivale a un conjunto de registros cuyo número es variable, dependiendo de cómo se ha definido la tabla de parámetros del disco. Por lo tanto, con una operación física de lectura se consigue leer el contenido de varios registros a la vez. Esta característica redunda en beneficio de la velocidad de acceso a disco, ya que los mecanismos de lectura son bastante lentos en comparación con otros componentes del sistema.

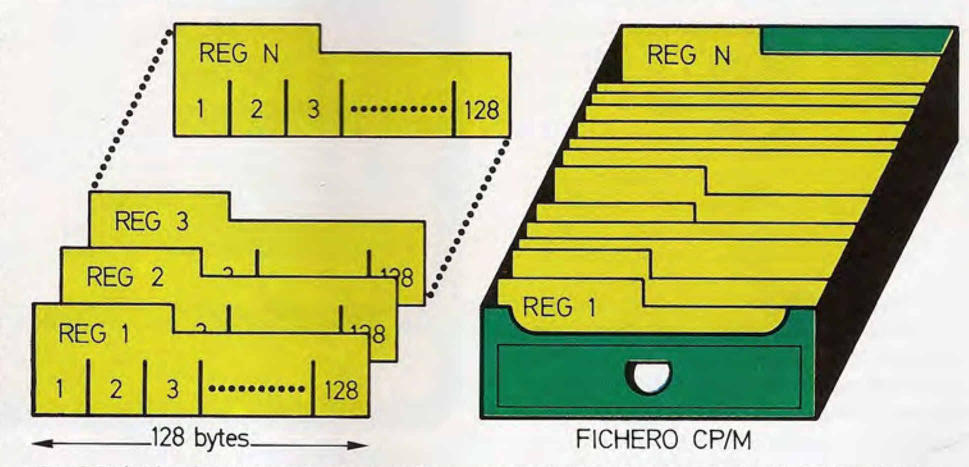


La misión del módulo BDOS es ofrecer al usuario los medios adecuados para un manejo cómodo y eficaz de la información almacenada en los discos.

Al igual que ocurre con los registros, los bloques son también agrupables en conjuntos que reciben el nombre de *extensiones*. El tamaño de cada extensión es también variable: de 1 a 16 k.

Los dos métodos de acceso a disco más frecuentes en el CP/M son los denominados de *acceso secuencial* y de *acceso directo.*

En el primero es obligatorio ir leyendo los



En el CP/M los ficheros están constituidos por una colección de registros de 128 bytes cada uno. Cada fichero puede incorporar un máximo de 65.536 registros.

registros, uno tras de otro, en el orden estricto en el que están colocados en el fichero. Ello significa que para leer un determinado registro del fichero, será necesario leer previamente todos los que le preceden.
El acceso directo obvia esta dificultad, puesto que permite acceder a cualquier registro sin tener que pasar necesariamente por los anteriores; desde luego, hay que saber cuál es la posición que ocupa el registro en cuestión.
La información completa de un fichero está almacenada en una sección de la memoria que el CP/M reserva para este cometido. Esta zona, denominada *bloque de control de fichero* (del inglés "File Control Block") dedica un espacio de 33 bytes de longitud para los ficheros de acceso secuencial y de 36 para los de acceso directo. Cada uno de estos espacios o bloques de control, memorizan el número de disco en el que está el fichero, su nombre y tipo, la extensión que ocupa, la posición actual para leer y escribir, información acerca del emplazamiento de los registros en el fichero...
De igual forma que se tiene una información completa acerca de cada fichero, también se dispone de la misma información a nivel disco; el BDOS genera un *catálogo o directorio* con la información de todos los ficheros residentes en el disco sobre el que se está operando, hasta un máximo de 128.
Cada entrada del catálogo ocupa 32 bytes: los 16 primeros constituyen un reflejo de los 16 bytes iniciales del FCB, tal y como quedarán al cerrar el fichero; a su vez, los 16 últimos bytes representan el mapa de ocupación de espacio en disco por parte del fichero.

ORDENES DIRECTAS A LOS DISPOSITIVOS FISICOS

El módulo BIOS incorpora toda una serie de funciones de bajo nivel necesarias para la interacción de los programas del CP/M y BDOS con el soporte físico de la máquina. Estos indican cómo hay que acceder a los distintos dispositivos que conforman el sistema; por ejemplo, gestionan el desplazamiento de la cabeza del disco para que se mueva a través de su superficie.
El BIOS contiene, en definitiva, los programas de entrada/salida específicos para la configuración hardware adoptada. Generalmente estos programas son confeccionados por el fabricante del microprocesador; su diseño es muy crítico, debido a que cualquier ligero error hará que no funcione el sistema o que lo haga inadecuadamente.
Si se desea tener aceso a periféricos de tipo no estándar, es preciso modificar consecuentemente el programa o programas al efecto del módulo BIOS.

REFERENCIAS A UN FICHERO EN CP/M

La identificación de un fichero en el sistema operativo CP/M consta de una referencia al disco en el que reside, del nombre de dicho fichero y el tipo que le haya sido asignado. En definitiva, el formato general es:

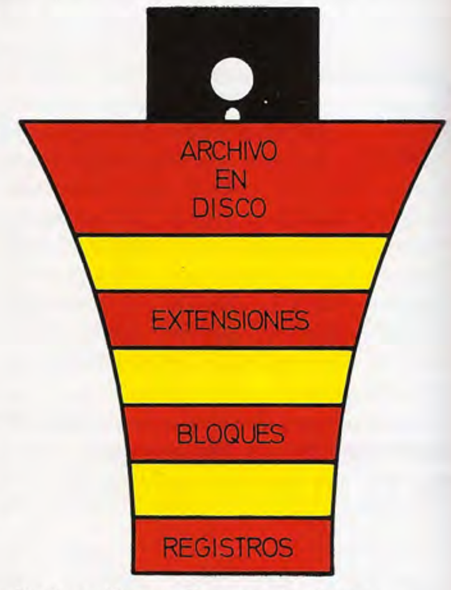
disco:nombre.tipo

Disco

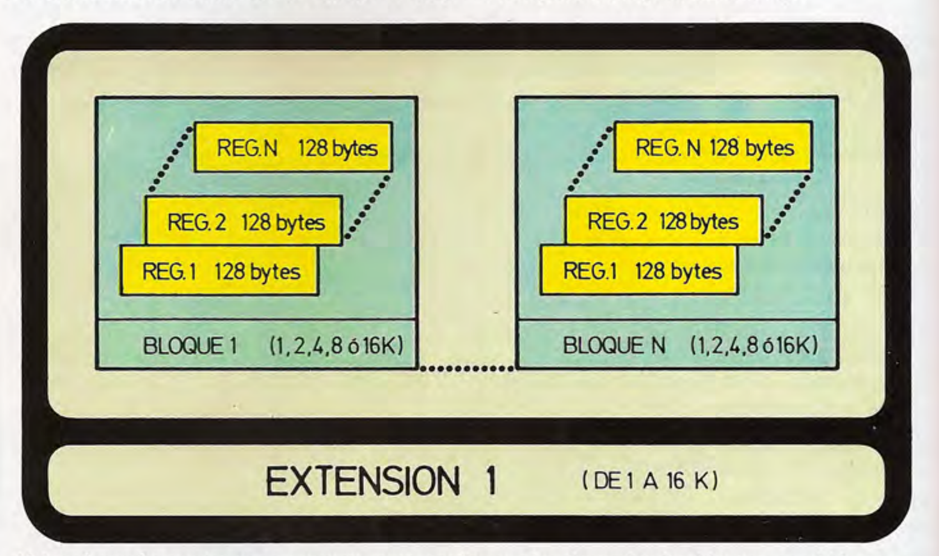
Los discos son referenciados por una letra que los identifica e indica al usuario cuál de ellos es accedido en un determinado instante. Habitualmente, es posible referenciar hasta un máximo de 16 discos.

Nombre del fichero

El nombre del fichero ha de contener un máximo de ocho caracteres alfanuméricos. Al efecto suelen utilizarse las letras mayúsculas; no obstante, en ciertos casos especiales es posible definir ficheros con letras minúsculas, e incluso sin nombre. Hay ciertas restricciones en las com-



Cada archivo en disco consta de una estructura en la que caben varios niveles. En la base están los registros, cuya asociación constituye un bloque. A su vez, la agrupación de varios bloques da cuerpo a las extensiones. Varias extensiones constituirán, en definitiva, el fichero en disco.



Al igual que los registros se agrupan para conformar los denominados bloques, también estos últimos son asociables, dando lugar a las "extensiones". El tamaño de cada extensión puede estar comprendido entre 1 y 16 K.

binaciones de caracteres capaces de generar el nombre de un fichero; determinados signos son inadmisibles y su empleo conduce a error. Los símbolos no utilizables en la combinación de un nombre de fichero coinciden con los que se indican a continuación:

< > [ ] ( ) * ? / ; , . :

La incorrección de los dos últimos caracteres es obvia: de utilizarlos confundirían al CCP, puesto que dicho módulo los reconoce como separadores. De igual forma, tampoco son válidos todos los caracteres de control, definidos por la combinación de la tecla CONTROL y otro carácter, además de aquellos caracteres que no tengan representación en la pantalla. A continuación se relacionan varios nombres de ficheros correctos e incorrectos:

NOMBRE:	Correcto.
Trabajo 1:	Correcto sólo en algunos casos (utiliza minúsculas).
NOTA*:	Incorrecto por contener un carácter no autorizado.
135910:	Correcto.
ESPECIFICACION:	Erróneo por tener más de 8 caracteres.
DATO<CTRL-U>:	Erróneo por incluir caracteres de control.

Manipulación y traslado de ficheros

Dentro del módulo CCP, existe un comando transitorio que permite el intercambio de información entre diversos periféricos del sistema. Dicho comando responde al nombre de programa de intercambio entre periféricos o PIP (del inglés: "Peripheral interchange program"). Las capacidades del PIP permiten copiar ficheros de un disco a otro o en el mismo disco, concatenar o unir varios ficheros distintos en un fichero único, convertir letras de mayúsculas a minúsculas al mismo tiempo que realiza la operación de copia...

El formato adecuado para la ejecución de este comando no es único, sino que admite dos variantes. En la primera de ellas, la línea de comando consta exclusivamente del comando en sí (PIP). Al procesarlo, la máquina pasa a instaurarse en un modo de trabajo o entorno de ejecución propio del comando (PIP); éste se caracteriza por la aparición en la pantalla de un indicador de petición, representado por un asterisco (*). En estas condiciones, el ordenador queda a la espera de recibir las acciones que el usuario desee llevar a cabo. Por supuesto, todas las operaciones que se ordenen deben ser de manipulación y traslado de ficheros: operaciones admisibles por el comando PIP. Tras introducir las operaciones pertinentes, debe cerrarse el comando PIP con una simple acción sobre la tecla de retorno (<CR>).

Un ejemplo ilustrativo del comando PIP, en el formato indicado, es el que refleja la siguiente pantalla:

```
A > PIP

* B:DATO.ASM=DATO.ASM
* B:NUEVO.TEX=B:VIEJO.TEX
* <CR>
A >
```

En él se ordena la realización de dos copias: una del fichero DATO.ASM del disco A al fichero DATO.ASM del disco B, y la otra del fichero VIEJO.TEX, ambos contenidos en el disco B. Como se observa, el cierre del comando se realiza con una acción sobre la tecla RETURN (retroceso del carro); el ordenador responde retornando a modo normal.

El segundo formato o modo de empleo del comando, es el de uso más frecuente. Una sola línea de comando incluye la orden PIP y define los ficheros implicados en la operación. Una vez ejecutada la línea en cuestión, el monitor vuelve a retomar el control de la máquina. A título de ejemplo, la orden siguiente realiza una copia múltiple con un sólo comando PIP:

A>PIP TODO.BAS=BAS1.BAS,BAS2.BAS,BAS3.BAS

Exactamente, ordena la copia de los ficheros BAS1.BAS,BAS2.BAS y BAS3.BAS,uno a continuación de otro, en un único fichero denominado TODO.BAS.

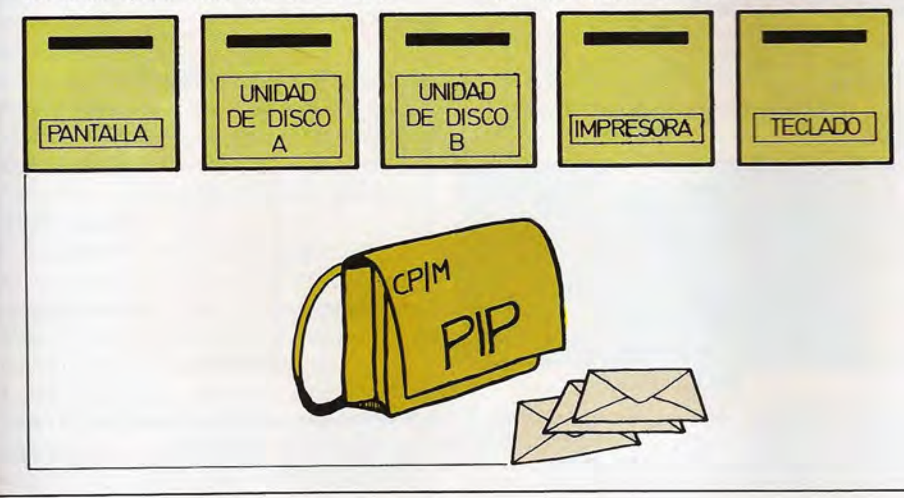
A raíz de los ejemplos precedentes se observa que el formato general bajo el que actúa el comando PIP adopta la siguiente forma:

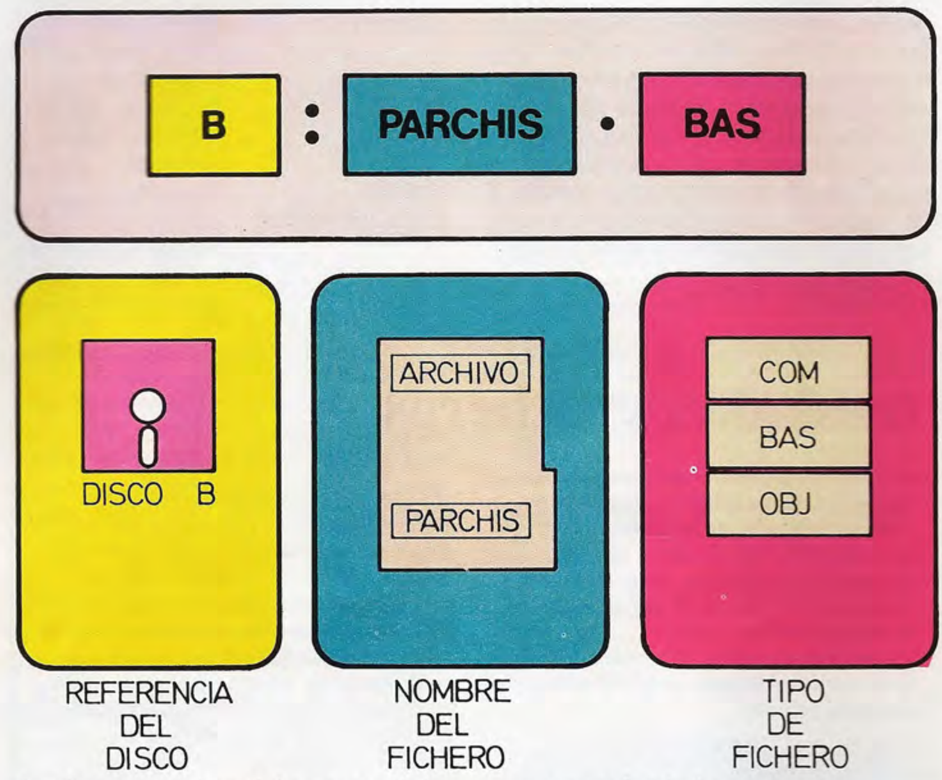
fichero de destino=fichero fuente 1, fichero fuente 2... (opciones)

El fichero de destino designa al fichero o periférico que debe recibir los datos; por su parte, la zona de fichero fuente define al fichero o ficheros que deben ser copiados en el de destino. A continuación, puede incluirse una zona de opciones, cuya misión es la de matizar las transferencias ordenadas.

Las diversas opciones disponibles permiten, entre otras cosas:

— Truncar o cortar un fichero a partir de un número de columna n.
— Tratar sólo ficheros que hayan sido modificados después de una fecha límite.
— Parar o comenzar la copia a partir del punto en el que se encuentren ciertas letras claves.
— Comprobar que la transacción se ha realizado correctamente.





La identificación de un fichero consta de las tres zonas que refleja la ilustración: una referencia al disco en el que reside, el nombre otorgado al fichero y una tercera etiqueta que indica el tipo de fichero.

Como norma general, es conveniente utilizar nombres de ficheros que sean significativos, de tal forma que constituyan un reflejo de su contenido y ayuden al usuario a su rápida identificación.

Tipo de fichero

Los diversos tipos de ficheros aluden a la función o al contenido de los mismos. La referencia al tipo, situada a continuación del nombre del fichero, consta de un máximo de tres caracteres; a su vez, es objeto de las mismas restricciones en su formación que las descritas anteriormente para los nombres de fichero.

El sistema reconoce varios tipos de ficheros estándar, lo cual no es óbice para que el usuario pueda crear tipos propios. Por ejemplo, en las diversas etapas del desarrollo de un programa se pueden utilizar tipos no estándar que indiquen cuál es el estado intermedio del programa; al concluir el desarrollo, puede ya referenciarse el fichero con el tipo que le sea propio. En la mayor parte de los casos no es necesario indicar el tipo de los ficheros, a no ser que el comando que le afecte exija esta condición.

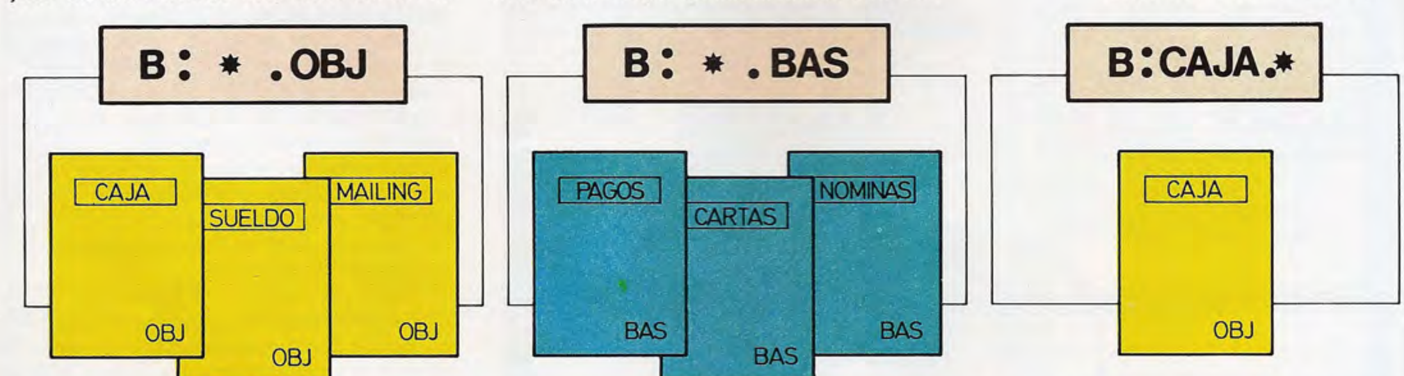
Entre los múltiples tipos de ficheros estándar aceptados por el CP/M, están, por ejemplo, los siguientes:

COM:	Comando ejecutable.
ASM:	Programa fuente en lenguaje ensamblador.
BAS:	Programa fuente en BASIC.
$$$:	Fichero temporal del editor.
OBJ:	Programa objeto resultado de un proceso de compilación.

A veces, es posible que no se conozca con exactitud el nombre del fichero que se desea utilizar; o también puede darse el caso que se pretenda utilizar más de un fichero simultáneamente. Para resolver ambas situaciones, el sistema operativo permite unos formatos especiales llamados *referencias ambiguas* o "wildcards" que son capaces de generalizar el significado del comando.

Cuando se utiliza el asterisco (*) éste sustituye al nombre y/o tipo del fichero de forma completa. A su vez, el signo de interrogación (?) puede reemplazar a uno de los caracteres del nombre o tipo de fichero. Los siguientes ejemplos ilustran el empleo de las referencias ambiguas.

*.FOR:	Se refiere a todos los ficheros FORTRAN.
NUEVO.*:	Alude a todos los ficheros de cualquier tipo, cuyo nombre es NUEVO.
.:	Referencia a todos los ficheros.
???1.FOR:	Se refiere a todos los ficheros cuyo nombre posea cuatro caracteres, siendo el último de ellos el número 1, y cuyo tipo sea FORTRAN.



El sistema operativo CP/M admite el uso de referencias ambiguas (wildcards) en la definición de los ficheros. Por ejemplo, el asterisco (*) es el signo adecuado para generalizar los campos de nombre y tipo del fichero. En el gráfico, el asterisco se utiliza para referenciar a todos los ficheros de tipo OBJ resistentes en el disco B (izquierda), a todos los de tipo BAS residentes en el mismo disco (centro) y a los ficheros de cualquier tipo, contenidos en el disco B, cuyo nombre es caja (derecha).

Wordstar (I)

Una aplicación estandarizada para el tratamiento de textos

En el presente capítulo se abre el estudio de las principales aplicaciones creadas para el proceso de textos en ordenadores personales. El primero de los paquetes, cuyo análisis práctico ocupará los tres próximos capítulos de la obra, es el WORDSTAR; sin duda alguna, el tratamiento de textos más extendido y popular en la categoría de equipos. Más adelante, se acometerá el estudio de las aplicaciones de proceso de textos predominantes en el marco del ordenador personal.

¿QUE ES EL WORDSTAR?

Micropro International Corporation es la firma americana creadora de este paquete para el tratamiento de textos destinado al ámbito de los ordenadores personales. En la actualidad existen versiones de WORDSTAR creadas para su compatibilidad con distintos sistemas operativos; no obstante, la versión inicial y más difundida se desarrolló para trabajar bajo el control del sistema operativo CP/M.

Resulta difícil definir en una única frase qué es el WORDSTAR. En una breve síntesis cabe catalogarlo como "una aplicación interactiva dedicada al proceso de palabras con objeto de producir un documento escrito".

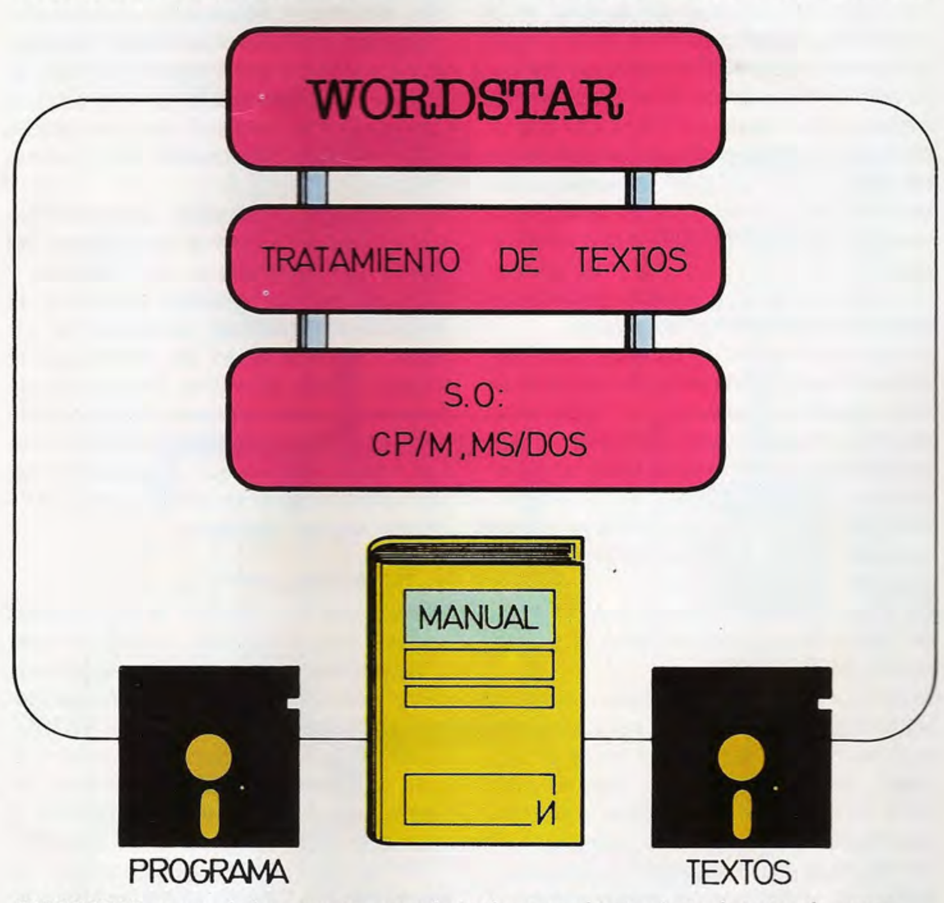
Su funcionamiento exige una configuración en la que el ordenador personal esté complementado con algunos periféricos básicos: pantalla, teclado, una unidad para discos flexibles y una impresora.

El proceso típico permite introducir un texto original a través del teclado. Este se almacenará en un disco flexible para, posteriormente, visualizarlo y/o modificarlo sobre la pantalla, antes de imprimirlo con un determinado formato. En todo caso, los archivos que puede tratar no tienen por qué ser cargados de forma manual; otras aplicaciones, independientes del WORDSTAR aunque capaces de crear archivos compatibles, pueden generar el archivo en disco. Posteriormente, podrá utilizarse dicho archivo para realizar un tratamiento de textos sobre su contenido (modificar el formato de salida, variar el contenido, obtener copias nominales...).

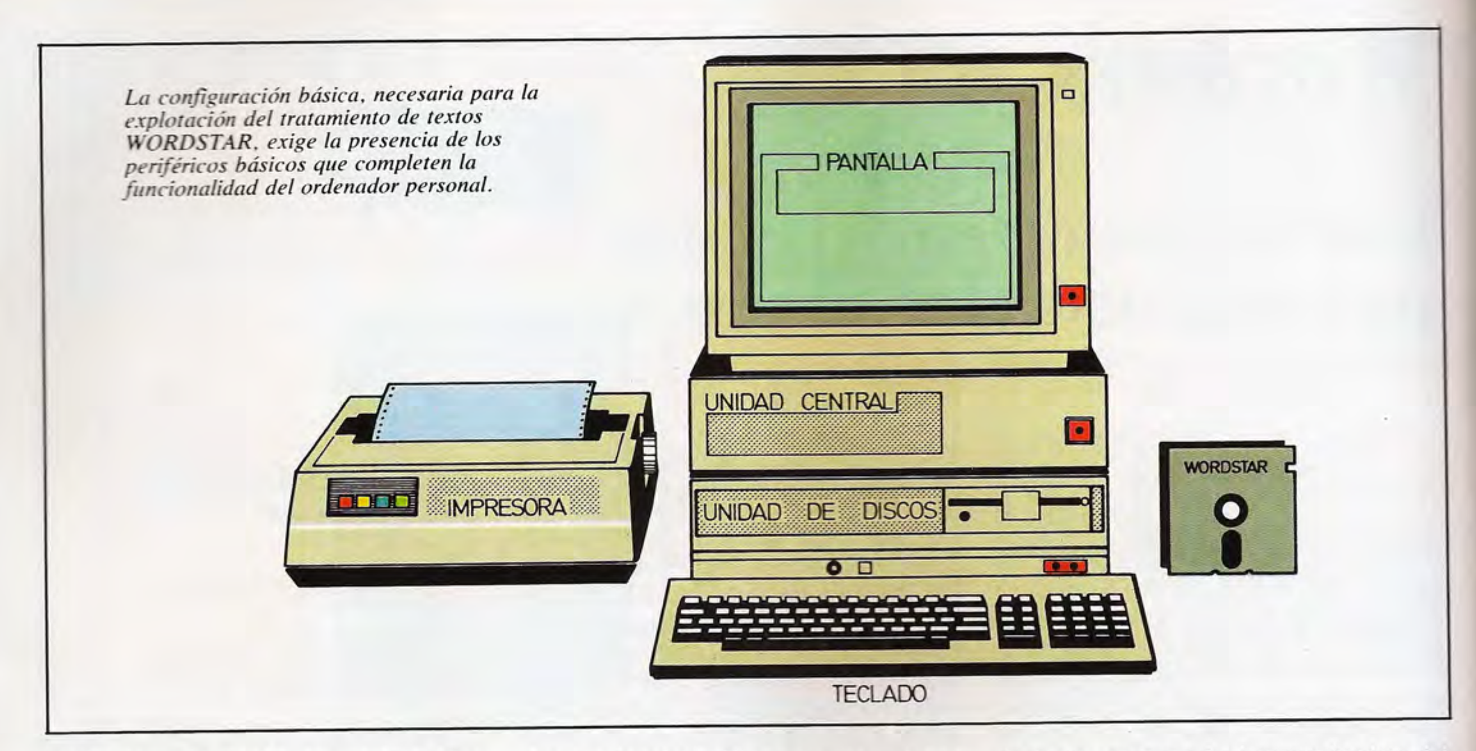
PRINCIPALES COMANDOS DEL WORDSTAR

1. COMANDOS BASICOS DE EDICION EN PANTALLA

La misión encomendada a los comandos de edición, tanto en este procesador de



WORDSTAR es uno de los paquetes estandarizados para el tratamiento de textos de mayor difusión en el ámbito de los ordenadores personales. Su autoría corresponde a la firma americana Micropro International.



La configuración básica, necesaria para la explotación del tratamiento de textos WORDSTAR, exige la presencia de los periféricos básicos que completen la funcionalidad del ordenador personal.

textos como en todos los que se analizarán en otros capítulos de la obra, es la creación y modificación de documentos almacenados en un dispositivo de memoria auxiliar, habitualmente en disco flexible o rígido. Algunas de las principales operaciones realizables mediante los comandos de edición son: creación de ventanas de edición, de tal forma que en una misma pantalla física se puedan editar simultáneamente varios documentos, formateado de textos en la pantalla, solicitud de ayuda para la inclusión de guiones...

Al utilizar el editor de pantalla, ésta presentará siempre una parte del documento que se está introduciendo o modificando en ese preciso instante. Cualquier actualización que se realice en el texto, se reflejará de forma instantánea en la pantalla; para ello, es preciso posicionar el cursor (posición activa de la pantalla) en el lugar deseado antes de escribir.

En el teclado se encuentran grupos varios de teclas fundamentales para el trabajo con el WORDSTAR.

En el apartado de edición, cabe destacar a un primer grupo de teclas, marcadas normalmente con una flecha hacia arriba, abajo, derecha o izquierda, cuya función es la de mover el cursor sobre la pantalla sin modificar los caracteres sobre los que se desplaza. Por supuesto, el otro grupo básico de teclas son las que constituyen el teclado alfanumérico; zona coincidente con el teclado de una máquina de escribir convencional. En él se encuentran las teclas alfabéticas, numéricas, el espaciador y algunos símbolos y caracteres especiales. Cuando se pulsa alguna de ellas, el carácter correspondiente aparece sobre el cursor; éste se desplaza hacia su derecha para permitir la introducción de un nuevo carácter.

En definitiva, el teclado alfanumérico constituye la vía para la introducción del texto; una tarea que se verá apoyada y facilitada por las funciones del editor. El propio editor permite comprobar el aspecto final del texto, de forma que el usuario pueda corregirlo, en el caso de cometer un error o por un simple cambio de criterio, antes de plasmarlo en el papel.

Las modificaciones que la aplicación realiza automáticamente sobre el texto introducido son las siguientes:

— *Fin de línea automático*

Inicialmente, los párrafos se escriben sin accionar la tecla RETURN, esto es, sin que el usuario se preocupe de cómo termina cada línea de texto. Cuando una palabra rebasa el margen derecho, el WORDSTAR la traslada sistemáticamente a la línea siguiente, inserta blancos para alinear el párrafo y muestra en la pantalla el resultado final. Por lo tanto, la tecla RETURN tan sólo se utilizará para señalar un fin de párrafo. De esta forma, el operador de WORDSTAR queda liberado de la acción de delimitar los finales de línea y, por lo tanto, no necesita levantar la vista del original que esté mecanografiando.

— *Gestión del texto*

Al utilizar la opción *fin de línea automático*, cada vez que se escriba una línea ésta quedará justificada entre los márgenes izquierdo y derecho. Desde luego, el usuario puede ordenar un ajuste distinto; en tal caso puede optar por otras alternativas que ofrece la aplicación: margen derecho irregular, escritura a doble o triple espacio, o centrado entre márgenes derecho e izquierdo distintos a los existentes.

— *Reforma de párrafos*

Para modificar cualquier párrafo del documento, el WORDSTAR ofrece un nutrido repertorio de comandos; comandos que permiten cambiar los márgenes, la separación entre líneas, pasar de margen derecho alineado a margen derecho irregular (o viceversa) o eliminar a voluntad caracteres, palabras o líneas del párrafo en curso de modificación.

2. COMANDOS ESPECIALES DE EDICION EN PANTALLA

Además de los comandos básicos de edición, apuntados en el apartado precedente, el WORDSTAR dispone de comandos especiales para ordenar diversas funciones, por ejemplo:

— Colocar o quitar tabuladores para el texto.
— Copiar, borrar o mover bloques completos del texto.
— Colocar o buscar marcas dentro del texto.
— Buscar porciones del texto.
— Buscar y modificar porciones del texto.
— Insertar dentro de un documento el texto de otro documento archivado en la memoria de masa.

También dispone de un sistema de ayuda para la colocación de guiones en palabras cortadas. De hecho, el WORDSTAR determina automáticamente los lugares adecuados para dividir una palabra entre líneas, utilizando un guión. Por supuesto, el operador puede decidir la oportunidad de colocar o suprimir la presencia del guión. Los guiones insertados por el programa tienen carácter temporal, ya que si se realiza una reforma posterior del texto, el guión podría quedar ubicado en mitad de la línea; de ahí que su escritura quede condicionada a cada situación momentánea.

La edición de texto empieza con la escritura del mismo en la pantalla, utilizando al efecto el teclado incorporado al ordenador. Para situar el punto de escritura en cualquier zona de la pantalla, el usuario cuenta con cuatro teclas cuya misión es desplazar el cursor en los cuatro posibles sentidos (hacia arriba, abajo, izquierda y derecha).



Elementos software para el proceso de textos

Al hablar de los elementos hardware necesarios para el proceso de textos, citábamos en primer lugar al propio ordenador; más exactamente, a la unidad central de la máquina en la que se realiza el tratamiento de textos. Tal decisión queda justificada por el hecho de que el resto de los componentes físicos son incapaces de funcionar sin la intervención del procesador. Pues bien, el propio procesador (elemento físico, al fin y al cabo) es incapaz de realizar ningún tipo de tarea sin la intervención de programas que lo instruyan correctamente. En síntesis, el software imprescindible para el funcionamiento de un sistema para el tratamiento de textos consta de dos tipos de programas.

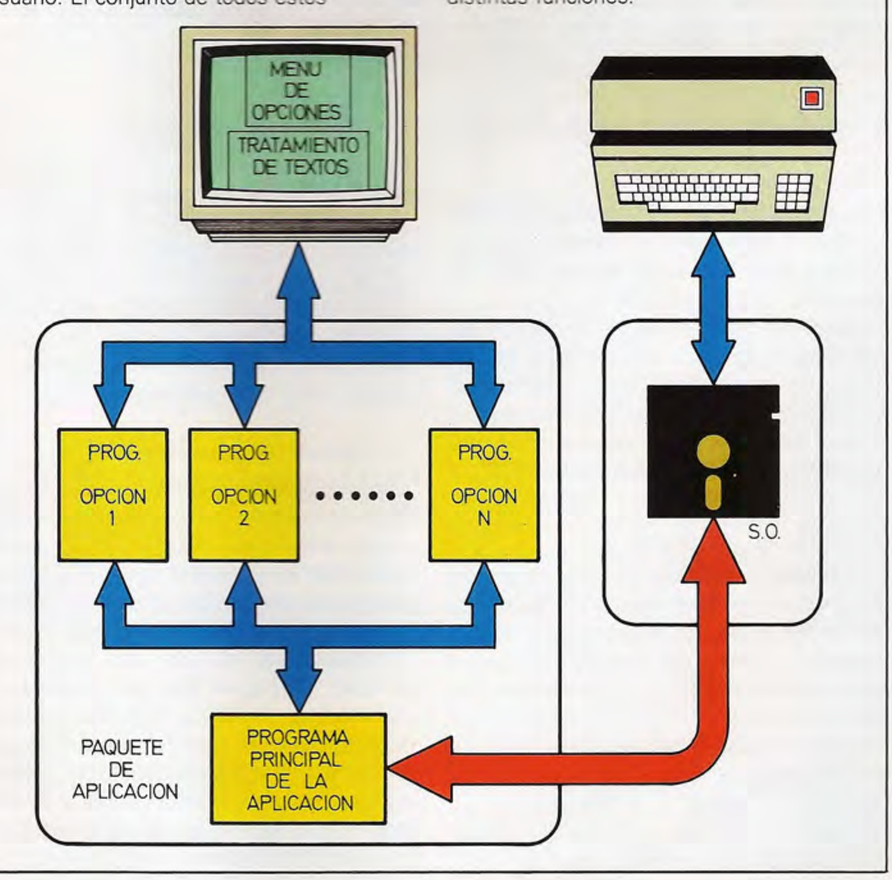
- El sistema operativo

Así se denomina al conjunto de programas que permiten el funcionamiento general del ordenador; tanto para su empleo como sistema dedicado al tratamiento de textos, como para su aplicación en cualquier otro tipo de actividad.

- Paquete de programas para el tratamiento de textos

Se encarga de instruir a la máquina para que ésta brinde al usuario todo un abanico de facilidades que permitan realizar distintas operaciones sobre el texto inicial. Su misión puede sintetizarse afirmando que admite como entrada de información un texto escrito, sin preocupaciones estéticas, y produce como salida el mismo texto pero con una presentación impecable, definida por el usuario.

Los paquetes de aplicación para el tratamiento de textos son, habitualmente, modulares. Incluyen un conjunto de programas especializados en la gestión de las distintas opciones que se ofrecen al usuario. El conjunto de todos estos programas está gestionado por otro programa denominado principal; éste suele presentar un menú de opciones que, seleccionadas por el operario, dan paso a la ejecución de los adecuados programas especializados en las distintas funciones.





Los comandos especiales de edición, permiten introducir el texto original con comodidad y rapidez. A título de ejemplo, la figura muestra una secuencia de "búsqueda y modificación" de palabra, asociada a uno de los comandos especiales disponibles en el WORDSTAR.

— *Efectos especiales*

El WORDSTAR está preparado para imprimir textos con palabras subrayadas, en negrita, cursiva, etc. Para ello es necesario utilizar un carácter especial precediendo a la palabra afectada; este carácter será quien active automáticamente el estilo de la presentación.
Además de los ya citados, los efectos especiales más interesantes son: subíndices, superíndices, espaciado proporcional entre los caracteres de una palabra y separación variable entre líneas.
También es posible combinar los efectos especiales. Por ejemplo, se puede escribir una misma palabra subrayada, en negrita y como subíndice.

— *Justificación del texto con micro-espacios*

Para lograr una mejor distribución de los caracteres, el espacio en blanco entre palabras se distribuye de tal forma que la separación entre las distintas palabras de una línea y la separación entre los distintos caracteres de una palabra ofrezca un aspecto agradable. Si se utiliza una impresora de calidad estos ajustes se realizarán mediante micro-espacios. Sin embargo, cuando la impresora utilizada no permite el empleo de micro-espacios, el ajuste se realizará en base a espacios completos.

3. COMANDOS DE IMPRESION

Este grupo de comandos del WORDSTAR facilitan la impresión de documentos ya escritos y almacenados con la ayuda de los comandos de edición.
La elección del formato de impresión sobre el papel es el cometido básico de este grupo de comandos. Las principales características de formato seleccionables a la hora de imprimir un documento son las que se relacionan a continuación:

— *Formateo de página*

En el margen superior de cada página del documento y/o en el margen inferior pueden incluirse de forma automática "cabeceras" o "pies" de página, sin que el usuario deba introducir su contenido más que una vez.
Para determinar el tamaño de la página y los márgenes se utilizan los denominados "comandos punto". Si el operador no utiliza estos comandos, el WORDSTAR tomará valores por defecto.

— *Numeración de páginas*

El número de cada página puede escribirse en el pie de página, centrado a la derecha o a la izquierda. No obstante, si el usuario lo desea, también podrá situarse en la parte superior de cada página. La numeración puede realizarla el programa automáticamente, si bien, el usuario siempre puede optar por modificarla o incluso omitir su presencia.

— *Control del salto de página*

Los cambios de página se realizan a medida que éstas se completan. En todo caso, el salto de página también puede realizarse en cualquier situación, activado por los comandos de fin de página. Estos comandos pueden ser incondicionales o condicionales; los primeros provocan el cambio de página sea cual fuere la circunstancia, mientras que los segundos sólo ejecutan el salto de página si se cumple una determinada condición (generalmente, se utilizan para asegurar la escritura de un mínimo grupo de líneas dentro de la misma página).

4. COMANDOS DE OPERADOR

Para concluir con la presentación de los distintos grupos de comandos del WORDSTAR, cabe mencionar a los comandos destinados a activar las funciones de impresión:
— Comienzo o final de numeración.
El usuario puede seleccionar la numeración de las páginas ya sea prefijando el número de la primera o de la última página de texto. La propia aplicación se encargará de otorgar a cada página el número adecuado, dependiendo de la condición expresada por el operador.
— Utilización de caracteres para salto de página.
— Detención entre páginas, para colocar hojas sueltas en la impresora, en el caso de no utilizar papel continuo.
— Impresión del contenido de un archivo sin formato de página (impresión del texto tal y como se introdujo en el equipo).
— Impresión o almacenamiento del documento en un archivo en lugar de realizarlo a través de la impresora.

Operadores de relación

Comparando datos. Los comandos NEW, STOP y CONT

En el capítulo precedente se introdujo el concepto de ruptura de la secuencia de ejecución de un programa. En efecto, tal ruptura se traducía en un salto o bifurcación dentro del programa; éste podía ser incondicional o condicional. Esta última posibilidad, sintetizada en las instrucciones de tipo IF/THEN/ELSE, suponía la entrada en escena de una condición cuyo cumplimiento desviaba la secuencia de ejecución hacia otra zona del programa en curso.

Cabe recordar que la instrucción IF/THEN/ELSE consta de tres zonas bien diferenciadas: la condición a chequear localizada tras la palabra clave IF, la instrucción o instrucciones a ejecutar si se cumple dicha condición (zona THEN), y la instrucción o instrucciones a ejecutar en caso contrario (zona ELSE).

No cabe duda que la parte principal de la instrucción la constituye la condición impuesta, puesto que de su cumplimiento depende que se realice una u otra acción. Esta condición será, en general, una expresión que una vez evaluada producirá un resultado del tipo *cierto* o *falso*. Por ejemplo, en el caso que sigue:

```
IF A=0 THEN PRINT "A VALE CERO"
```

la acción indicada por el comando PRINT se ejecuta si es cierto que la variable A toma el valor cero. El ordenador, internamente, es capaz de evaluar la expresión situada entre las palabras IF y THEN. El resultado de esta evaluación será un dato de tipo "lógico", cuyo valor sólo puede ser *cierto* o *falso*.

COMPARANDO DATOS

Las expresiones que se utilizan habitualmente para imponer una condición suelen ser comparaciones entre datos. En la mayoría de los casos interesa ejecutar órdenes distintas en función del valor de una determinada variable. Por ejemplo: calcular la raíz cuadrada de una variable sólo cuando ésta almacena un valor positivo y eludir el cálculo si el valor es negativo:

```
IF A>0 THEN RAIZ=SQR (A)
```

Es evidente, pues, que se hace necesario contar con un método que permita formular adecuadamente las comparaciones entre diferentes datos.

Esta comparación se codifica en BASIC por medio de los siguientes operadores de relación:

=, <, >, < >, <=, >=

Los operadores de relación se aplican entre dos datos, constantes o variables, dando como resultado un valor de tipo lógico, esto es: respondiendo con *cierto* o *falso*.

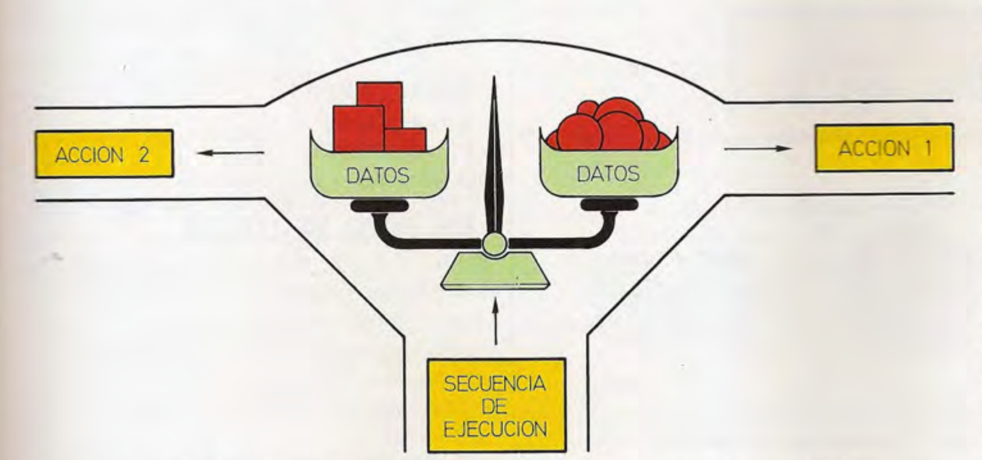
El operador "=" evalúa la igualdad entre los datos especificados, de tal forma que si son iguales la expresión adoptará el valor *cierto*, y si son distintos el valor *falso*.

```
3=3 ...CIERTO
3=4 ...FALSO
5=2 ...FALSO
```

El operador "<" asignará el valor *cierto* a la expresión en la que se encuentre cuando el dato situado a su izquierda sea estrictamente menor que el dato colocado a su derecha. Si el valor de la izquierda es mayor o igual que el valor de la derecha, la expresión tomará el valor *falso*.

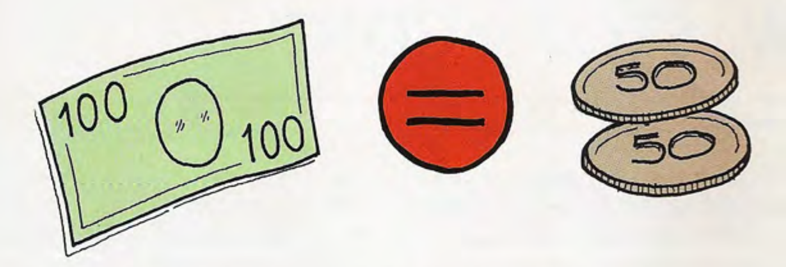
```
3<4 ...CIERTO
3<3 ...FALSO
5<2 ...FALSO
```

El signo ">" indica "mayor que"; es análogo al anterior, si bien, en este caso, el operando de la izquierda debe ser mayor que el de la derecha para que el valor obtenido sea *cierto*.



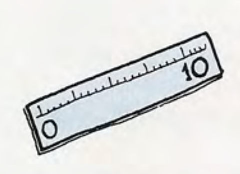
La secuencia de ejecución de los programas puede sufrir alteraciones, de acuerdo al cumplimiento o no de condiciones impuestas por el programador. Estas suelen adoptar la forma de comparaciones entre datos establecidas por medio de operadores de relación.

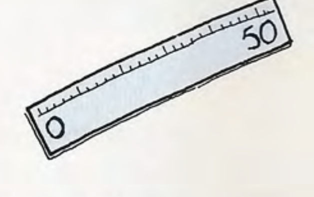
IGUAL QUE



El operador "igual que" (=) evalúa la igualdad entre los dos datos colocados uno a cada lado del operador.

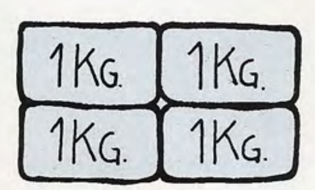
MENOR QUE

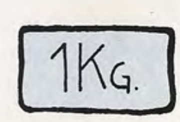


La condición "menor que" (<) responde con "cierto" (1 lógico) cuando el dato situado a la izquierda del mismo es inferior al situado a la derecha del operador.

MAYOR QUE



La tercera de las condiciones básicas se concreta en la relación "mayor que" (>); la respuesta es "cierto" cuando el dato situado a la izquierda del operador es de superior magnitud al de la derecha.

3>3 ...FALSO
3>4 ...FALSO
5>2 ...CIERTO

Los operadores definidos se complementan con otros tres que realizan funciones contrarias.

El operador complementario de la igualdad es el de desigualdad, cuyo signo es "<>". Cuando los datos colocados a izquierda y derecha del referido signo sean distintos, el resultado de la comparación será *cierto*.

3< >3 ...FALSO
3< >4 ...CIERTO
5< >2 ...CIERTO

En definitiva: A<>B será *cierto* siempre que A=B sea *falso* y viceversa.

A su vez, los operadores mayor y menor también disponen de sus respectivos complementarios. Fijémonos en el primero: ¿Cuándo no será cierto A > B? Naturalmente, cuando A sea inferior a B. Pero ¡cuidado!, también cuando A y B sean iguales. El operador capaz de evaluar esta condición se denomina "menor o igual" y su signo característico es "<=".

3<=3 ...CIERTO
3<=4 ...CIERTO
5<=2 ...FALSO

El último operador de relación es el complementario de "menor que" (<). Este debe ser *cierto* cuando el resultado de "<" sea *falso*. Análogamente al caso anterior existen dos posibilidades: que el primer dato sea mayor que el segundo o que ambos sean iguales. Por lo tanto, este operador se identificará como "mayor o igual". Su correspondiente signo es ">=", como era de esperar.

3>=3 ...CIERTO
3>=4 ...FALSO
5>=2 ...CIERTO

UN POCO DE LOGICA

El resultado de evaluar la condición impuesta en una instrucción IF, es un dato de tipo lógico. Sabemos que el ordenador

trabaja internamente con datos expresados a base de ceros y unos. Estos no son ni más ni menos que los mismos datos introducidos, aunque codificados de forma que sean inteligibles para la máquina. Por suerte, el programador no tiene por qué conocer esta compleja representación interna de los datos. Cabe, sin embargo, señalar un punto relacionado con el tema que nos ocupa: la evaluación de condiciones IF. El resultado de dicha evaluación —lo que hemos dado en llamar dato de tipo lógico—, admite dos valores, a saber: *cierto* o *falso*. Estos se codifican en el ordenador en forma de "1" si la respuesta es *cierto*, ó "0" en el caso contrario *(falso)*.

Desde luego, estos datos pueden ser almacenados en variables numéricas, lo que se consigue mediante una sentencia de asignación, en la cual la expresión a evaluar irá entre paréntesis. Por ejemplo:

```
POSITIVO=(A>=0)
```

Si el contenido de la variable A es un número positivo, o incluso el valor 0, la variable POSITIVO adoptará el valor *cierto* (1), en el caso contrario adoptará el valor *falso* (0).

Las variables que contienen datos de tipo lógico, resultantes de evaluar una expresión, pueden utilizarse como condiciones dentro de una instrucción IF. Por ejemplo:

```
10 IF POSITIVO THEN ...
```

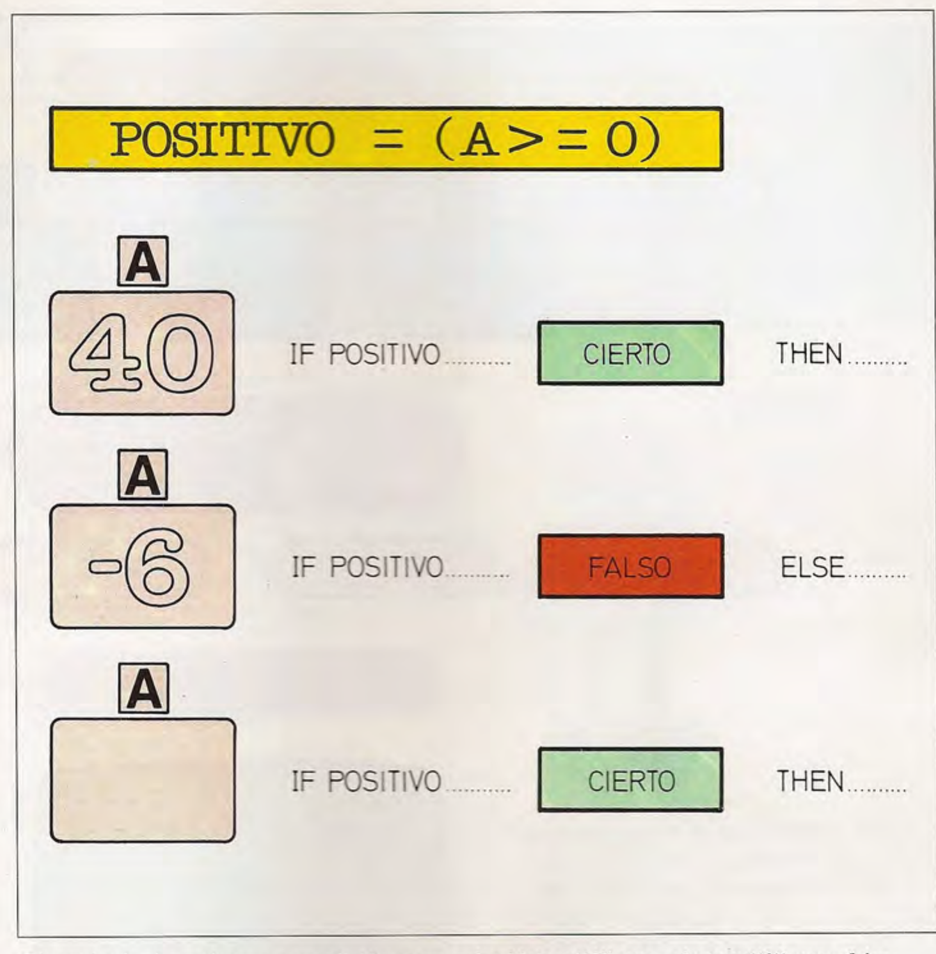
En este caso, POSITIVO debe contener un dato de tipo lógico (*cierto* o *falso:* 1 ó 0 en su expresión binaria).

Empleando este método, se pueden simplificar las expresiones de la condición dentro de las instrucciones IF/THEN/ELSE. Así, por ejemplo, es posible introducir condiciones como la que sigue:

```
10 IF A THEN ...ELSE ...
```

en donde A es una variable numérica. En este caso, cuando la variable A adopte el valor 0, por efecto de la ejecución del programa, se ejecutará la zona ELSE de la instrucción. Por el contrario, cuando tome el valor 1 u otro cualquiera distinto de 0, se ejecutarán las instrucciones localizadas en la zona THEN, al igual que si la variable A tuviera en ese instante el valor lógico *cierto*.

Así pues, es perfectamente factible utilizar una variable numérica como condición en la sentencia IF. Esta es una alternativa



El resultado de evaluar una condición es un valor lógico: "1" para cierto y "0" para falso. Este puede almacenarse en una variable numérica y ser utilizado dentro de instrucciones de bifurcación condicional.
En el ejemplo, la variable POSITIVO —cuyo valor queda establecido en la asignación inicial—, se utiliza para imponer la condición en la instrucción IF/THEN/ELSE.

particularmente útil cuando se desea realizar una acción siempre que determinado dato *no* sea nulo. Por ejemplo, si se trata de evitar que el ordenador realice una división por cero, puede adoptarse la siguiente instrucción:

```
50 IF DATO THEN SOLU=100/DATO
```

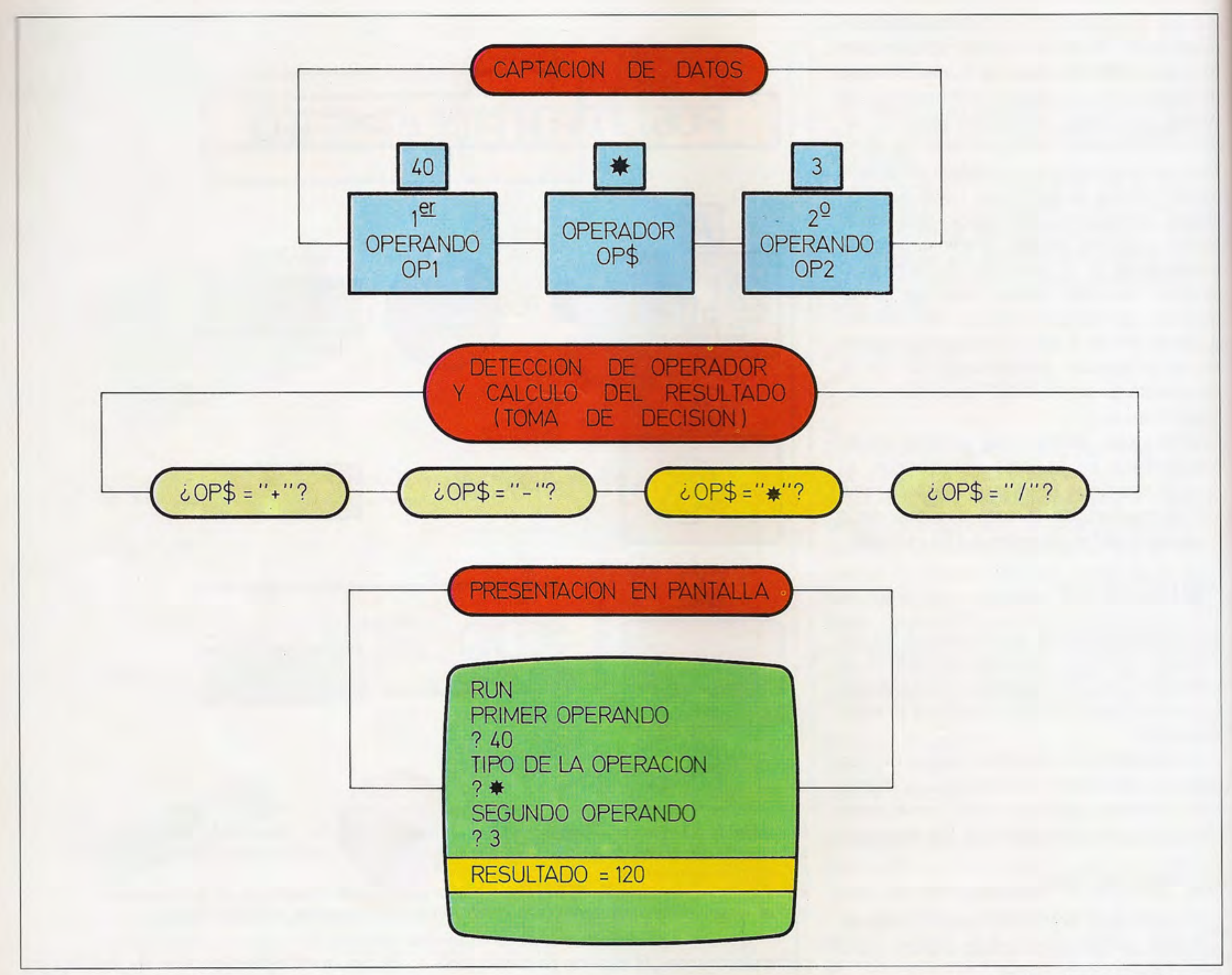
El cociente (zona THEN) sólo se ejecutará cuando DATO sea cierto. Como quiera que se trata de una variable numérica, sólo se tomará como falsa cuando su valor sea cero. Por lo tanto, la división se realizará únicamente si DATO es distinto de cero... precisamente, lo que se pretendía.

No hay que perder de vista que la referida instrucción es totalmente análoga a:

```
50 IF DATO < > 0 THEN
   SOLU=100/DATO
```

El siguiente ejemplo pone en práctica los conceptos explicados. El programa calculará la raíz cuadrada del número introducido siempre y cuando éste sea positivo. Si se introduce un número negativo, el ordenador lo rechazará y solicitará la introducción de otro número (la raíz cuadrada de un número negativo es una operación errónea).

```
10 INPUT A
20 LET POSITIVO=(A>=0)
30 IF POSITIVO THEN GOTO 50
40 GOTO 10
50 LET B=SQR(A)
60 PRINT "LA RAIZ CUADRADA
   DE ";A; "ES: ";B
70 END
■
```



La figura refleja el funcionamiento del programa BASIC, incluido en el texto, adecuado para convertir al ordenador en una calculadora. Las tres zonas básicas del programa se concretan en la captación de los datos, en la detección del operador y cálculo del resultado, y en la presentación del mismo en la pantalla.

En la línea 20 se asigna un valor lógico a la variable POSITIVO. Esta variable se utiliza posteriormente como condición en la línea 30.

```
RUN
?4
LA RAIZ CUADRADA DE 4 ES: 2
RUN
?0
?25
LA RAIZ CUADRADA DE 25 ES: 5
■
```

UNA CALCULADORA A SU SERVICIO

A modo de resumen práctico de las instrucciones de salto condicional, vamos a construir un programa BASIC que convierta al ordenador en una calculadora; elemental pero adecuada para resolver las cuatro operaciones aritméticas. Su empleo se reducirá a introducir el primer operando, tras éste el tipo de operación y por último el segundo operando.

El primero de los operandos se capta a través de la instrucción INPUT OP1. La variable OP1 será la encargada de guardar el valor del primer operando.

El siguiente paso es introducir el tipo de operación que se desea realizar. Para no complicar el programa, las posibilidades se reducen a las cuatro operaciones básicas. Mediante la instrucción INPUT OP$, se lee el operador y se almacena en la variable: OP$. Por último, la instrucción INPUT OP2 se ocupará de leer el segundo operando que pasará a constituir el contenido de la variable OP2.

¿Qué hay que hacer para que el ordenador ejecute la operación solicitada? Muy fácil: sencillamente, poner en práctica las posibilidades de las instrucciones IF/THEN/ELSE. Estas deben ir comprobando las distintas operaciones y al coincidir el signo de operación con el contenido de OP$, ejecutar la referida operación con los datos OP1 y OP2. El resultado se guardará en una nueva variable cuyo nombre es RESUL.
Una vez realizado el cálculo, hay que presentar el resultado en la pantalla, por medio de una instrucción PRINT. Finalmente, la última línea del programa ordena un salto incondicional al principio del mismo; ello hará posible realizar sucesivos cálculos sin tener que ordenar una nueva ejecución.

```
  5 LET RESUL=0
 10 PRINT "PRIMER OPERANDO"
 20 INPUT OP1
 30 PRINT "TIPO DE LA OPERACION"
 40 INPUT OP$
 50 PRINT "SEGUNDO OPERANDO"
 60 INPUT OP2
 70 IF OP$= "+" THEN LET
    RESUL=OP1+OP2
 80 IF OP$= "−" THEN LET
    RESUL=OP1−OP2
 90 IF OP$= "*" THEN LET
    RESUL=OP1*OP2
100 IF OP$= "/" THEN LET
    RESUL=OP1/OP2
110 PRINT "RESULTADO="; RESUL
120 GOTO 10
```

Y SI YA NO ES UTIL...

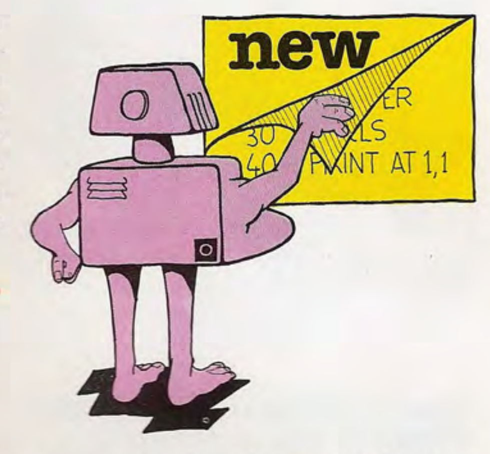
Sin duda alguna, el programa no será de mucha utilidad para quien posea una calculadora de bolsillo; por lo tanto, será necesario borrarlo de la memoria del ordenador, donde ha estado almacenado hasta ahora.
La solución drástica es, desde luego, desconectar la alimentación de la máquina. Este es un método nada recomendable; existen procedimientos más "elegantes" para borrar un programa. Uno de ellos es utilizar el comando NEW (nuevo). La ejecución de esta orden como instrucción directa, origina el borrado completo de todas las líneas del programa que hubiera en la memoria del ordenador, así como de todas las variables inicializadas y utilizadas hasta el momento. Además, el sistema queda inicializado a la espera de que un nuevo programa sea introducido a través del teclado, o por otro medio.

NEW

Borra el programa que se encuentra en la memoria del ordenador, así como todas las variables definidas hasta el momento.

Formato: NEW

Ejemplo: NEW

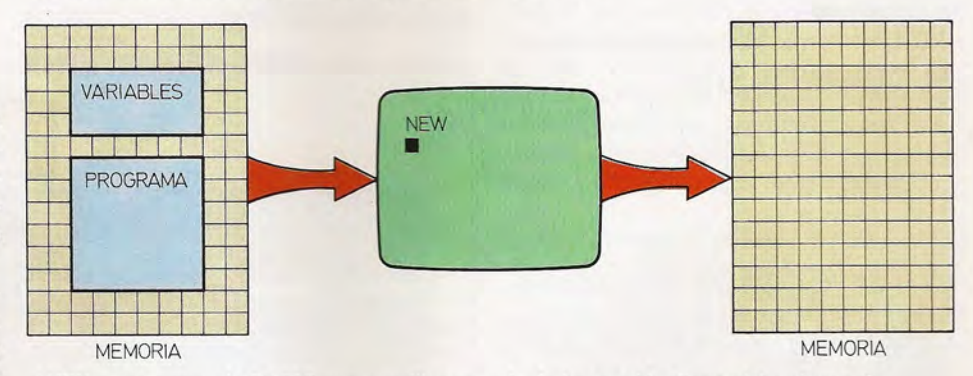


INTERRUPCION DE UN PROGRAMA

La ejecución de un programa evoluciona de forma secuencial y ordenada, comenzando por la línea de programa cuyo número es inferior. A partir de ésta se procede con la siguiente, respetando siempre el orden de menor a mayor en los números de las líneas ejecutadas.
Cuando el ordenador encuentra una instrucción END se detiene la ejecución. Esta instrucción suele colocarse en la última línea del programa. No obstante, nada impide situarla en cualquier otro punto del mismo. En tal caso, las líneas siguientes no se ejecutarán en la secuencia lógica, ya que el ordenador interpreta que el programa finaliza en la instrucción END.
El siguiente programa incluye dos líneas tras la instrucción END que no serán ejecutadas:

```
10 PRINT "LINEA 10"
20 PRINT "LINEA 20"
30 END
40 PRINT "LINEA 40"
50 PRINT "LINEA 50"

RUN
LINEA 10
LINEA 20
```



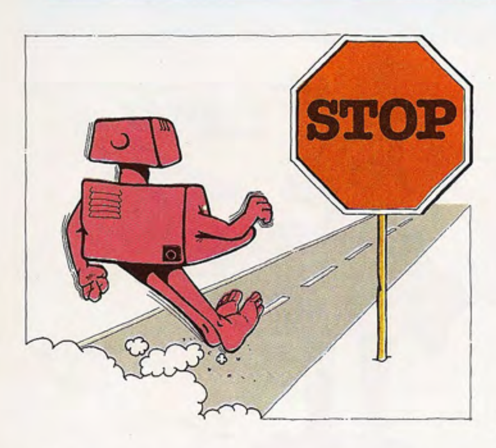
NEW es un comando BASIC que se utiliza a modo de instrucción directa (sin número de línea). Su cometido es borrar de la memoria del ordenador el programa BASIC y las variables almacenadas en ella.

STOP

Detiene la ejecución del programa en curso.

Formato: (Número de línea) STOP

Ejemplo: 50 STOP



El hecho de colocar una instrucción END en medio de un programa puede parecer absurdo, no obstante, llega a ser útil en ciertos casos; por ejemplo, si se desea interrumpir la ejecución bajo ciertas condiciones. El fragmento de programa que sigue dará por terminada la ejecución cuando la variable X tome el valor cero.

```
...
120 IF X < > 0 THEN GOTO 140
130 END
140 REM CONTINUACION
...
```

Las líneas de la 140 en adelante se ejecutarán exclusivamente cuando, al pasar por la comparación, X resulte distinto de 0. Si, por cualquier motivo, X fuera siempre nulo, las referidas líneas no se ejecutarían en ningún caso.

EL COMANDO STOP

En el BASIC existe otro comando destinado a interrumpir la ejecución; éste es el comando STOP. Al igual que END, puede situarse en cualquier punto del programa; e incluso puede reemplazar al propio END. Por ejemplo:



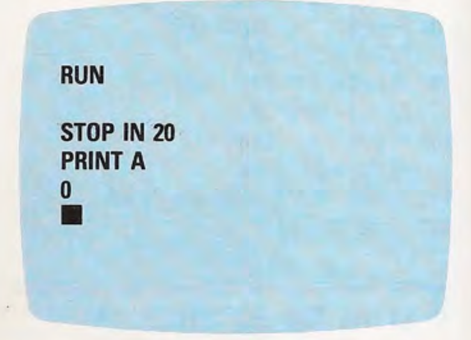
Una vez modificada la línea 30, puede ordenarse la ejecución del programa, lo que conducirá al siguiente resultado en la pantalla:

```
RUN
LINEA 10
LINEA 20
STOP IN 30
■
```

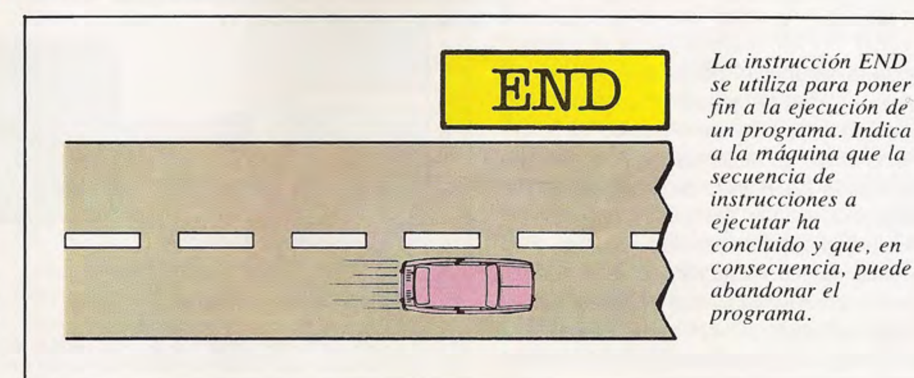
La primera diferencia que se aprecia al comparar ambas ejecuciones reside en el mensaje final. Al detenerse la ejecución, el comando STOP muestra un mensaje de parada (Stop, en inglés), en el que se indica la línea del programa en la que se ha producido la interrupción. Esta característica puede resultar útil a la hora de depurar un programa, colocando instrucciones STOP en puntos claves del mismo. En el siguiente ejemplo, el programa se interrumpe en varios puntos para, más tarde, visualizar en modo directo el valor de cierta variable.

```
10 LET A=0
20 STOP
30 LET A=A+1
40 STOP
50 IF A=1 THEN LET A=2
60 STOP
70 GOTO 30
■
```

Al lanzar la ejecución, el programa se detiene en la línea 20. En ese instante puede examinarse el valor actual de A utilizando una instrucción PRINT en modo directo:

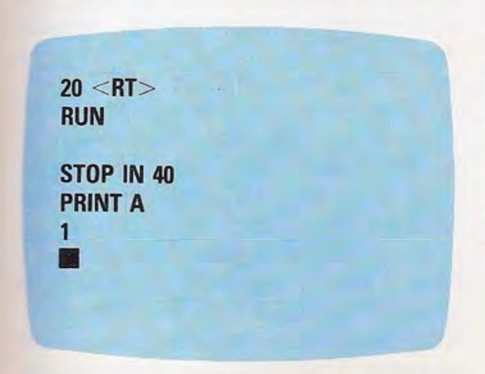


A continuación, una vez comprobado el valor de A, puede retirarse del programa la línea 20. Ello permitirá que el programa llegue a ejecutar la línea 30.



La instrucción END se utiliza para poner fin a la ejecución de un programa. Indica a la máquina que la secuencia de instrucciones a ejecutar ha concluido y que, en consecuencia, puede abandonar el programa.

TABLA DE CONVERSION			
ORDENADOR	NEW	STOP	CONT
	NEW	STOP	CONT
APPLE II (APPLESOFT)	NEW	STOP	CONT
APRICOT (M-BASIC)	NEW	STOP	CONT
ATARI	NEW	STOP	CONT
CBM 64	NEW	STOP	CONT
DRAGON	NEW	STOP	CONT
EQUIPOS MSX	NEW	STOP	CONT
HP-150	NEW	STOP	CONT
IBM PC	NEW	STOP	CONT
MPF	NEW	STOP	CONT
NCR DM-V (MS-BASIC)	NEW	STOP	CONT
NEW BRAIN	NEW	STOP	CONT
ORIC	NEW	STOP	CONT
SHARP MZ-700 (MZ-BASIC)	NEW	STOP	CONT
SINCLAIR QL	NEW	STOP	CONTINUE
SPECTRAVIDEO	NEW	STOP	CONT
ZX-SPECTRUM	NEW	STOP	CONT



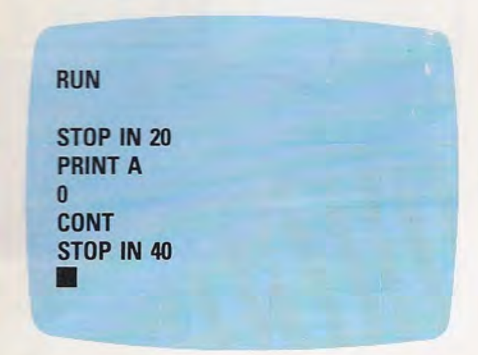
La operación anterior (borrado de la línea y nueva ejecución) puede reiterarse hasta comprobar que el valor de la variable es el correcto a lo largo de la ejecución del programa. También es posible chequear y alterar cualquier otra característica que varíe en el transcurso de la ejecución.
El ejemplo propuesto revela como única diferencia entre el uso de STOP o END la indicación, en el primer caso, de la línea en la que se produce la interrupción. Sin embargo, STOP tiene otra peculiaridad que lo hace más potente. Esta característica está íntimamente relacionada con un nuevo comando: CONT.

EL COMANDO CONT

De nuevo, vamos a repetir el ejemplo anterior, aunque esta vez utilizando el comando CONT para reanudar la ejecución del programa interrumpido por medio de STOP.
CONT ha de utilizarse siempre en modo directo. Una vez introducido y tras accionar la tecla RETURN, la ejecución CONTinuará en la línea siguiente a aquella en la que se detuvo.
Con el programa inicial, sin eliminar las instrucciones STOP, el resultado sería el siguiente:



La ejecución se ha detenido en la línea 20. Al igual que en el caso anterior se ha verificado el valor de la variable A. No obstante, ahora se reactiva la ejecución introduciendo la orden CONT:



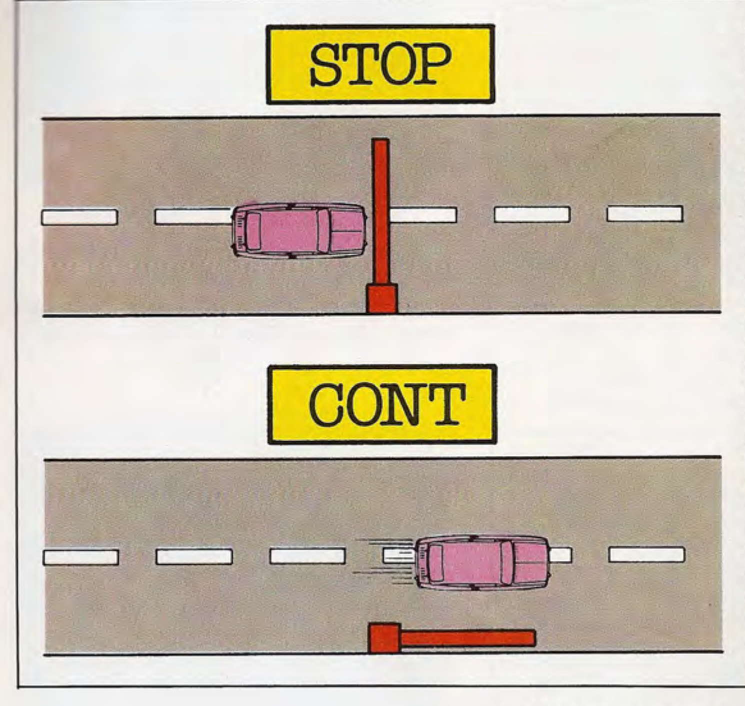
De nuevo puede visualizarse el contenido actual de A. Cabe observar que en este caso no ha sido necesario borrar la línea 20 para, posteriormente, reiniciar la ejecución. CONT permite continuar en la siguiente línea, en este caso la 30. Como es natural, el contenido actual de las variables no se ve alterado por el uso de STOP y CONT.
Con la ayuda de estos dos comandos es posible realizar un seguimiento del programa, interrumpiéndolo en el punto deseado y ordenando más tarde su reanudación.
Ambos comandos pueden también utilizarse para otros menesteres. Por ejemplo, a la hora de detener la ejecución del programa para apuntar datos intermedios. En ocasiones y dada la rapidez de cálculo del ordenador, los datos pueden aparecer en pantalla a muy alta velocidad; es entonces cuando se aconseja el uso de

CONT

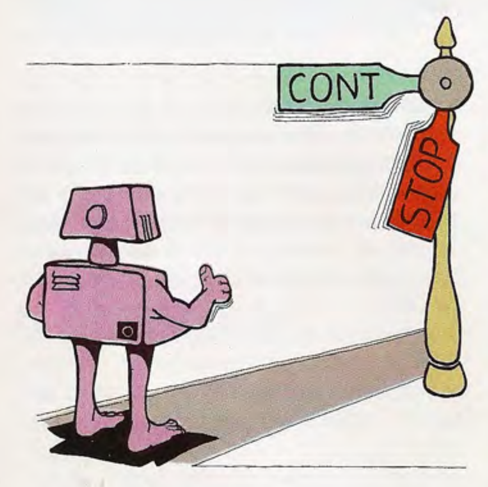
Reanuda la ejecución de un programa detenido por efecto del comando STOP.

Formato: CONT

Ejemplo: CONT



A diferencia con el comando END, STOP no supone la conclusión definitiva del programa, sino que sólo interrumpe la ejecución del mismo. Algo semejante a la colocación de una barrera en la trayectoria de un automóvil (secuencia de ejecución del programa). Por supuesto, una vez levantada la barrera por efecto del comando CONT, seguirá el avance del automóvil (continuará la ejecución del programa.



STOP. El siguiente ejemplo es un programa capaz de mostrar en pantalla los múltiplos de 11 a partir de 1000.

```
10 LET MUL=1001
20 PRINT MUL
30 LET MUL=MUL+11
40 GOTO 20
```

El bucle creado por medio de la instrucción GOTO 20 es muy rápido. En pocos instantes, la pantalla aparecerá repleta de múltiplos de 11; los primeros irán desapareciendo por la parte superior de la misma. Ello dificulta, cuando no hace imposible, su anotación.

Desde luego, existen otros métodos para "congelar" la presentación en pantalla, sin embargo, el más directo e inmediato es el que recurre al uso de STOP. Por ejemplo:

```
10 LET MUL=1001
20 PRINT MUL
25 STOP
30 LET MUL=MUL+11
40 GOTO 20
```

Ahora, la ejecución del programa se detendrá después de mostrar cada número sucesivo. Ello permitirá al usuario anotarlos, uno a uno, e introducir la orden CONT para pasar al siguiente dato.

El programa que sigue realiza la misma acción, si bien, su ejecución sólo se detendrá tras presentar cada grupo sucesivo de 20 números.

```
 10 REM MULTIPLOS DE 11
 20 LET MUL=1001
 30 LET C=0
 40 CLS
 50 PRINT "MULTIPLOS DE 11"
 60 PRINT
 70 PRINT MUL
 80 LET MUL=MUL+11
 90 LET C=C+1
100 IF C < 20 THEN GOTO 70
110 STOP
120 GOTO 30
```

Las líneas 20, 70 y 80 actúan de forma similar a las del programa anterior. En este segundo programa la clave se encuentra en las líneas 90 y 100. La primera de ellas contabiliza en la variable C los números presentados. La línea 100, por su parte, evalúa la expresión C < 20, que será cierta mientras la cantidad de números visualizados sea inferior a 20. En dicha situación, el programa salta a la línea 70, imprimiendo el siguiente múltiplo.

La ejecución continúa con normalidad y de forma análoga hasta que se alcanza la cantidad de 20 datos en pantalla (C=20); instante en el que la condición deja de ser cierta y, en consecuencia, deja de producirse el salto a la línea 70. En su lugar, la ejecución prosigue en la línea inmediatamente posterior en la que se encuentra el comando STOP. Así pues, el programa se detiene únicamente tras presentar en pantalla un bloque de 20 datos.

El uso del comando CONT conduce a la reanudación del programa en la línea 120, de donde bifurcará hacia la 30. Esta última reinicia el contador C a cero, con lo que el programa queda en disposición de presentar un nuevo grupo de veinte múltiplos de once.

OPERADORES DE RELACION

OPERADOR	RELACION
=	Igual
<	Menor que
>	Mayor que
< >	Distinto
< =	Menor o igual que
> =	Mayor o igual que

Logo (6)

Operadores de identificación. El espacio de trabajo

Además de las posibilidades de manipular palabras y listas, el lenguaje LOGO permite realizar comparaciones e identificaciones de tipo. Los operadores integrados en este grupo, responden con TRUE (cierto) o FALSE (falso) como datos de salida. La utilidad real y práctica de estas salidas se verá en el capítulo dedicado al estudio de los *bucles*. Por ahora, su cometido exclusivo será identificar el tipo de dato con el que se está trabajando.
Un primer operador al respecto es WORDP. Este responderá TRUE en el caso de recibir como entrada una palabra y FALSE en caso contrario.
De forma análoga actúan LISTP y NUMBERP con listas y números, respectivamente.
Un buen ejercicio es combinar estos dos operadores con los estudiados en anteriores capítulos, para identificar el tipo de las salidas que entregan en cada situación. Por ejemplo:

```
PRINT WORDP [CURSO DE BASIC]
FALSE

PRINT WORDP FIRST [CURSO DE BASIC]
TRUE

PRINT LISTP BUTLAST LIST "DON :YO
TRUE
```

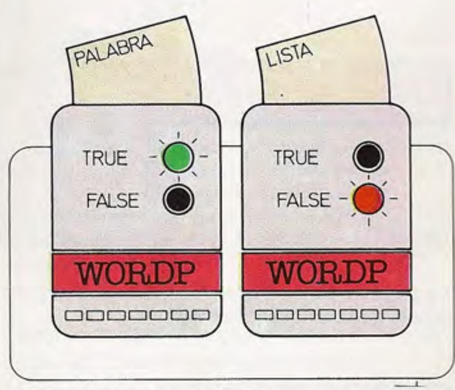
Tanto en el ámbito de las palabras como de las listas, cabe la nulidad; esto es: la palabra vacía ('') o la lista vacía ([ ]) en las que no interviene ningún elemento. El operador EMPTYP evalúa tal situación, y responde con TRUE si su dato de entrada coincide con una palabra o una lista vacía. Hay que tener en cuenta que para el LOGO una palabra vacía y una lista vacía no son la misma cosa. Ello puede comprobarse utilizando el operador EQUALP. EQUALP compara dos entradas y comunica si son o no iguales.

```
PRINT EMPTY [ ]
TRUE

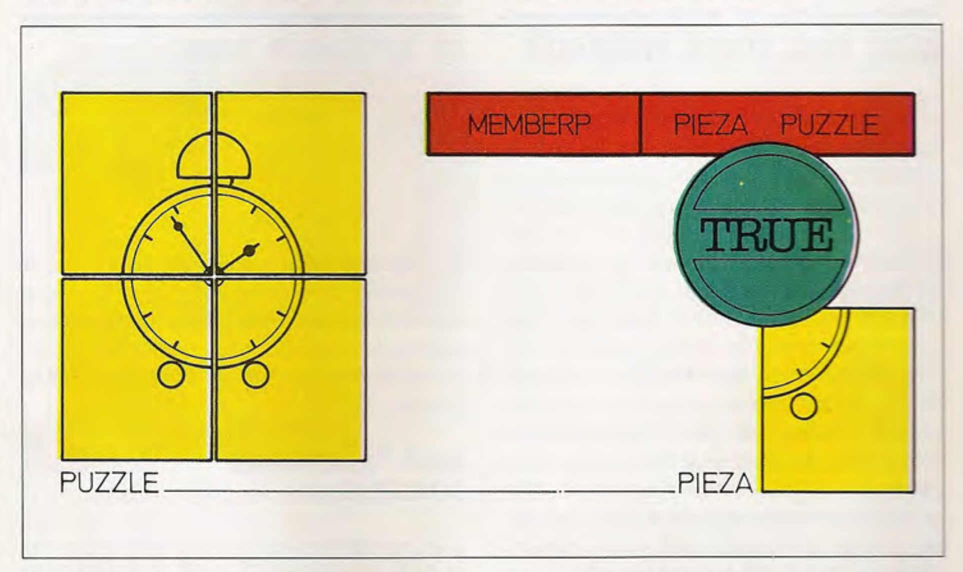
PRINT EQUALP" [ ]
FALSE

PRINT EQUALP "CASA [CASA]
FALSE
```

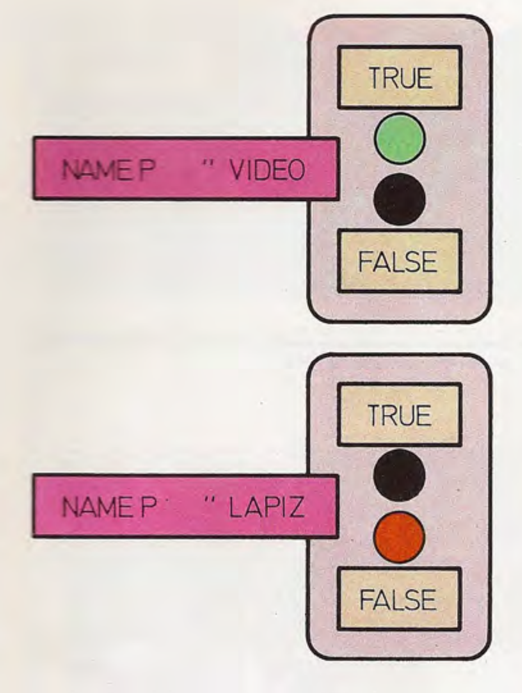
Un último operador de esta categoría es MEMBERP. Necesita dos datos de entrada: el primero puede ser una palabra, un número o una lista, mientras que el segundo debe ser obligatoriamente una lista.
La respuesta será TRUE si la primera entrada es un elemento que forma parte de la lista indicada. Es evidente que su utilidad se manifiesta a la hora de evaluar la presencia de una palabra o de una sublista dentro de una lista más compleja. Los siguientes ejemplos, basados en la LISTA3 definida, revelan su actuación.



Para evaluar el tipo de dato manipulado, el LOGO cuenta con tres operadores adecuados para detectar palabras, listas y números; éstos son, respectivamente: WORDP, LISTP y NUMBERP.



MEMBERP es otro de los operadores LOGO cuya respuesta es TRUE (cierto) o FALSE (falso). Su ejecución permite confirmar la presencia de una palabra o de una sublista dentro de la lista especificada.







La función del operador NAMEP es averiguar si una palabra coincide con el nombre de una variable previamente definida.

```
PRINT MEMBERP "EL :LISTA3
MAKE "LISTA3 [EL LENGUAJE LOGO]
TRUE

PRINT MEMBERP [EL] :LISTA3
FALSE

PRINT MEMBERP 7 [1 [3] 5 [7] 9]
FALSE
```

ALGO MAS SOBRE VARIABLES

Por el momento, el trabajo con variables se ha centrado casi por completo en el uso del comando MAKE, adecuado para asignar un dato a la variable especificada. Las posibilidades de MAKE no terminan ahí. Cada vez que se ejecuta el comando MAKE acompañado de un nuevo nombre de variable, éste reserva espacio para esa variable.
Al respecto, cabe recordar que no es posible utilizar una variable sin antes definirla previamente por medio de MAKE.
El LOGO lleva buena cuenta de las variables definidas.

Un método para averiguar si una palabra coincide con el nombre de una variable lo aporta el operador NAMEP. Este admite un dato de entrada y responde con TRUE si el mencionado dato es el nombre de una variable previamente definida. Al igual que otros operadores similares, descritos en capítulos precedentes, su verdadera utilidad práctica saltará a la luz al abordar el estudio de los bucles.

EL OPERADOR THING

En algunos casos, es conveniente que el contenido de una variable coincida con el nombre de otra. Esta es una situación que queda ilustrada por el siguiente ejemplo. Una vez introducidas las dos instrucciones que siguen:

```
MAKE "NUMERO "DIEZ
MAKE "DIEZ 10
```

para acceder al número 10 es preciso recurrir al nombre de variable :DIEZ. Esta es una función realizable por medio de un nuevo operador: THING.

THING "DIEZ actúa exactamente igual que :DIEZ, accediendo al número 10: THING admite como entrada la palabra que constituye el nombre de una variable y devuelve su contenido. No obstante, THING permite algo más.
En el mismo ejemplo, la salida de THING "NUMERO será "DIEZ; sin embargo, la salida de THING :NUMERO, coincidirá con 10. En definitiva, con THING es posible acceder al "contenido del contenido" de una variable.
Un ejemplo esclarecedor lo aporta el siguiente programa. Empieza con la asignación de dos variables (dos nombres a los que se asigna su correspondiente número de teléfono). A continuación, se define un procedimiento al que se asocia el parámetro :NOMBRE; un procedimiento que incluye la doble posibilidad de uso del operador THING: acompañado por un nombre de variable expresado como palabra ("NOMBRE) o como variable (:NOMBRE).

```
MAKE "LUIS 123 28 30
MAKE "PACO 430 26 26
TO TELEFONO :NOMBRE
PRINT THING "NOMBRE
PRINT THING :NOMBRE
END
TELEFONO DEFINED
```

La ejecución evidencia las dos posibles actuaciones del operador THING. Al ejecutar el procedimiento TELEFONO, dando como parámetro un nombre de variable, el ordenador responderá presentando el nombre en cuestión (PRINT THING "NOMBRE), seguido por el número de teléfono o contenido de la mencionada variable (PRINT THING :NOMBRE).

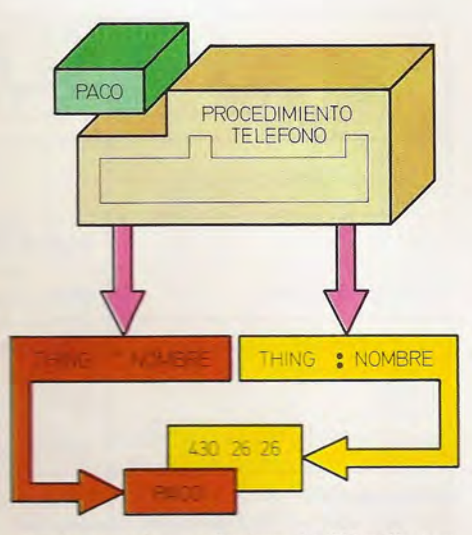
```
TELEFONO "LUIS
LUIS
123 28 30

TELEFONO "PACO
PACO
430 26 26
```

TABLA DE ORDENES-LOGO

INSTRUCCION	COMETIDO	OPERADOR/ COMANDO
WORD <objeto>	TRUE si <objeto> es palabra	Operador
LISTP <objeto>	TRUE si <objeto> es una lista	Operador
NUMBERP <objeto>	TRUE si <objeto> es un número	Operador
EMPTY <objeto>	TRUE si <objeto> vacío	Operador
EQUALP <obj1> <obj2>	TRUE si <obj1> y <obj2> son iguales	Operador
MEMBERP <obj> <lista>	TRUE si <obj> es elemento de <lista>	Operador
NAMEP <palabra>	TRUE si es nombre de variable	Operador
THING <objeto>	Devuelve el contenido	Operador
EDNS	Activa el editor de variables	Comando

<objeto> puede ser una palabra o una variable. <nombre> es el nombre del procedimiento (sin comillas). <variables> son las posibles variables locales (con dos puntos).



Al ejecutar el procedimiento "teléfono", cuyo parámetro de entrada es "nombre", se observa que el operador THING puede devolver tanto el nombre de la variable introducida como parámetro (THING "NOMBRE), como el contenido de la misma (THING :NOMBRE).

Obsérvese que THING "NOMBRE equivale a :NOMBRE y que THING :NOMBRE equivale a THING THING "NOMBRE.

VARIABLES GLOBALES Y LOCALES

Al hablar de procedimientos con parámetros se indicó que los parámetros de entrada actúan como variables asociadas a la ejecución del referido procedimiento. Estas variables son de un tipo especial y reciben el nombre de variables locales.
Las variables locales se diferencian de las restantes en varios aspectos.

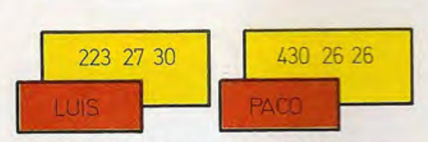
— No es preciso definirlas por medio del operador MAKE.
— Su nombre no es identificado por NAMEP.
— Sólo tienen validez dentro del procedimiento en el que se encuentran.

Las variables creadas con MAKE se denominan *variables globales.* Ello se debe a que pueden ser utilizadas por todos los procedimientos.
Las variables globales pueden ser visualizadas y modificadas por medio del *editor de variables,* al que se accede a través del comando EDNS (EDit NameS). El editor de variables funciona del mismo modo que el de procedimientos, con la salvedad de que también permite actualizar el valor de las variables globales.

ESPACIO DE TRABAJO

El *espacio de trabajo* (Work Space) es el conjunto de procedimientos y variables,





La función del operador THING es, en general, devolver el contenido. El procedimiento reflejado en la figura, asociado a las variables LUIS y PACO —cuyo contenido son los respectivos números de teléfono—, ilustra su doble funcionalidad.

definidos por el usuario, que se encuentran en la memoria del ordenador.
El lenguaje LOGO dispone de varios comandos para el manejo del espacio de trabajo.
En primer lugar están los comandos que muestran el contenido actual del espacio de trabajo. Todos ellos comienzan por PO (Print Out):

POTS (Print Out TitleS): muestra todos los nombres (títulos) de los procedimientos existentes en el espacio de trabajo.

POPS (Print Out ProcedureS): visualiza en la pantalla el contenido de todos los procedimientos.
Si lo que se desea es visualizar exclusivamente el contenido de un determinado procedimiento, hay que utilizar el prefijo PO seguido por el nombre del procedimiento. Por ejemplo:

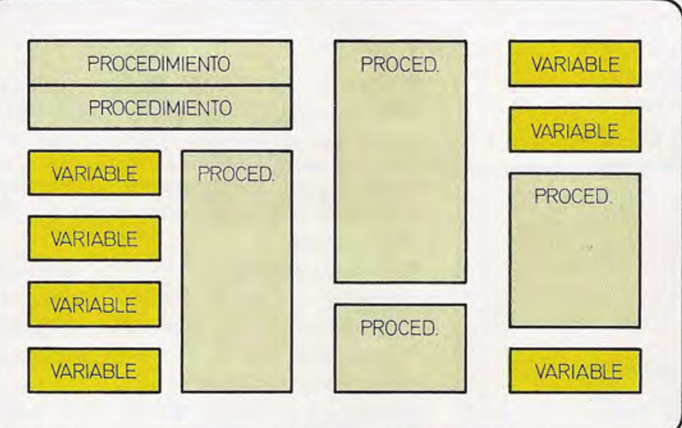
PO "INICIAL
mostrará en la pantalla el procedimiento INICIAL.

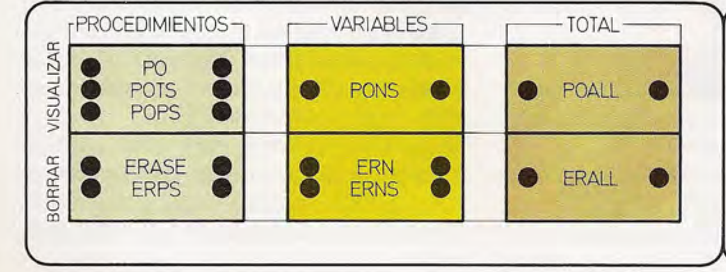
En general, los referidos comandos permiten ver, pero no modificar los procedimientos; para efectuar cualquier cambio es preciso recurrir al comando para la edición de procedimientos: EDIT.
Un segundo grupo de comandos especializados en el tratamiento del espacio de trabajo, está integrado por los destinados a visualizar las variables globales definidas.

ORDENES DE CONTROL DEL ESPACIO DE TRABAJO

INSTRUCCIONES	COMETIDO	OPERADOR/ COMANDO
ERALL	Borra el espacio de trabajo	Comando
ERASE <objeto>	Borra procedimiento(s)	Comando
ERN <objeto>	Borra variable(s)	Comando
ERNS	Borra todas las variables	Comando
ERPS	Borra todos los procedimientos	Comando
PO <objeto>	Muestra procedimiento	Comando
POALL	Muestra espacio de trabajo	Comando
PONS	Muestra variables (globales)	Comando
POPS	Muestra procedimientos (definiciones)	Comando
POTS	Muestra procedimientos (títulos)	Comando
NODES	Da el número de nodos libres	Operador
RECYCLE	Reorganiza memoria	Comando

<objeto> puede ser una palabra o una lista.





Ordenes LOGO para el control del "espacio de trabajo" o conjunto de procedimientos y variables, definidos por el usuario, que se encuentran en la memoria del ordenador.

PONS (Print Out NameS): muestra los nombres y contenidos de las variables; no obstante, al igual que PO y POPS, no permite su modificación.

POALL (Print Out ALL): este es otro comando afecto al espacio de trabajo; su especialidad es presentar en la pantalla el contenido de todo el espacio de trabajo.

Existe también la posibilidad de borrar una parte o la totalidad del espacio de trabajo. Los comandos utilizables para tal fin comienzan por ER (ERase).

ERASE: borra el procedimiento o procedimientos cuyos nombres acompañan al comando. Por ejemplo:

ERASE "CASA

hace que desaparezca del espacio de trabajo el procedimiento CASA.

Tal como se ha indicado, ERASE admite también una lista de nombres de procedimientos; todos ellos serán borrados por efecto del mismo comando.

ERPS (ERase ProcedureS): éste es un comando LOGO cuya especialidad es la de borrar, a la vez, todos los procedimientos que se encuentran en el espacio de trabajo.

Por lo que respecta a las variables, es posible borrar sólo algunas o todas a la vez. ERN borra la variable o lista de variables especificada, mientras que ERNS (ERase NameS) borra todas las variables de "un plumazo". Hay que señalar que ambos comandos eliminan tanto el contenido como el nombre de las variables, con lo cual, las variables borradas desaparecen por completo del espacio de trabajo.

Por último, cabe mencionar al comando ERALL (ERase All), cuya ejecución borra todo el espacio de trabajo.

Cada vez que se crea un procedimiento o variable queda ocupada una zona del espacio de trabajo. Para conocer cuánto espacio queda aún disponible se utiliza el operador NODES. Este responde con el número de "nodos" libres. Cada nodo equivale a cinco bytes de memoria.

Al eliminar procedimientos o variables se producen "huecos" en el espacio de trabajo ocupado. Para reordenar los procedimientos y variables hay que recurrir al comando RECYCLE. RECYCLE reúne los nodos libres reagrupando el espacio de trabajo utilizado. NODES proporciona el número real de nodos libres si se utiliza previamente RECYCLE. Un dato a señalar es que si no queda espacio suficiente a la hora de crear un procedimiento, se ejecuta automáticamente el comando RECYCLE.

Comandos básicos del CP/M

Comandos residentes y transitorios del sistema operativo CP/M

La misión habitualmente encomendada a un ordenador es la ejecución de programas de aplicación; ya sea confeccionados por el usuario, o bien realizados por compañías especializadas. En todo caso, cuando el ordenador queda liberado de la ejecución de los programas, ha de ser capaz de gestionar y mantener en orden al conjunto de información de tipo permanente puesto en juego por las aplicaciones: programas, ficheros de datos...

Para efectuar este control —tarea que cabría comparar con el mantenimiento y gestión de los archivos de una gran empresa—, el sistema operativo CP/M brinda al usuario toda una serie de herramientas en forma de comandos especiales. Estos comandos pueden dividirse, atendiendo al elemento sobre el que actúan, en dos grandes grupos:

- *Comandos que actúan sobre fichero o ficheros.*

Estos comandos consideran al fichero como una unidad de trabajo, y en su operación pueden copiar, eliminar o almacenar en memoria, pero siempre ficheros completos.

- *Comandos que actúan sobre la información contenida en un fichero.*

El miembro esencial de esta categoría es el editor del sistema operativo, con sus múltiples posibilidades de actuación.

Los comandos pueden también dividirse atendiendo a su situación en la memoria. Los comandos que "habitan" en la memoria central del ordenador y se cargan con el sistema operativo, se denominan *comandos residentes* o permanentes. A su vez, aquellos comandos que residen en un disco, junto al sistema operativo, y que son cargados cada vez que se van a emplear, reciben el nombre de *comandos transitorios.*

LOS COMANDOS RESIDENTES

Como ya se ha indicado, a esta categoría pertenecen los comandos CP/M, que son cargados en la máquina con el sistema operativo, y permanecen en la memoria central mientras no se desconecte el ordenador. Los más relevantes son los que se relacionan a continuación.

DIR

Su misión es mostrar el catálogo de los ficheros contenidos en un disquete. Adopta el siguiente formato:

A>*DIB x:*

El parámetro *x* representa el nombre de la unidad en la que se encuentra el disco del cual se desea obtener el catálogo de ficheros; puede ser un número o una letra, dependiendo del criterio de nomenclatura empleado. Este nombre puede omitirse, en cuyo caso se mostrará el catálogo de la unidad que el sistema considere por defecto (normalmente la principal del sistema).

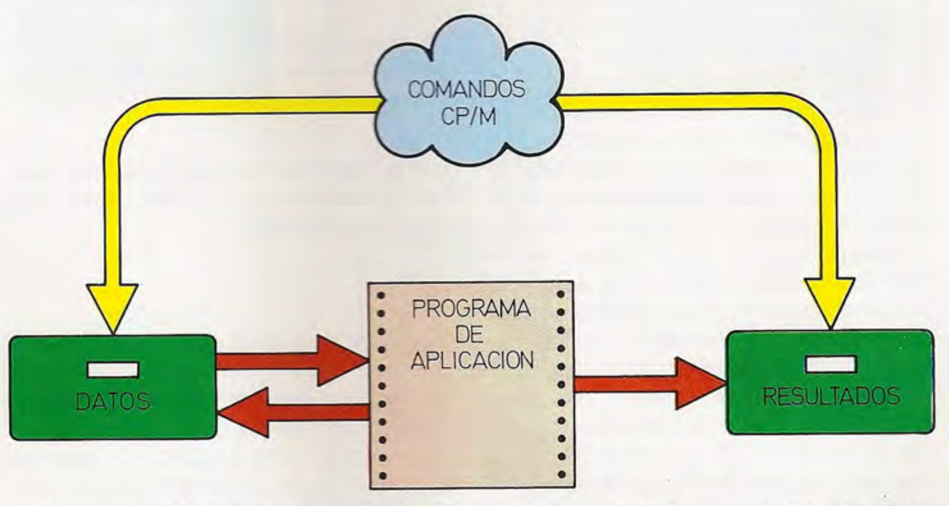
La orden admite también referencias ambiguas, lo que permite buscar los ficheros de un nombre específico. Por ejemplo:

A>*DIR B: *. BAK*

Su ejecución proporcionará el catálogo de todos los ficheros del tipo *.BAK* contenidos en el disco alojado en la unidad de lectura *B.*

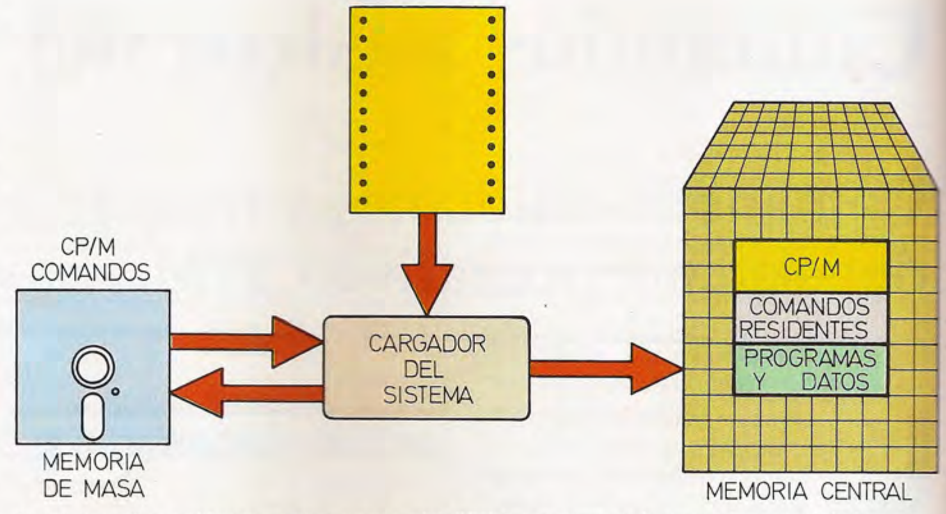
ERA

Su utilidad reside en el borrado de un fichero del disco. En la actualidad, el proceso que se sigue no es un borrado estricto del fichero, puesto que no se elimina la información; en realidad, se coloca un indicador en la cabecera del fichero, revelador de que los datos son obsoletos e inutilizables. Este espacio queda



Los comandos del CP/M actúan sobre los ficheros almacenados en la memoria del ordenador en cada instante. En consecuencia, para que sea posible utilizarlos con eficacia, es preciso trasladar los archivos a tratar desde el disco hasta la memoria central del equipo.

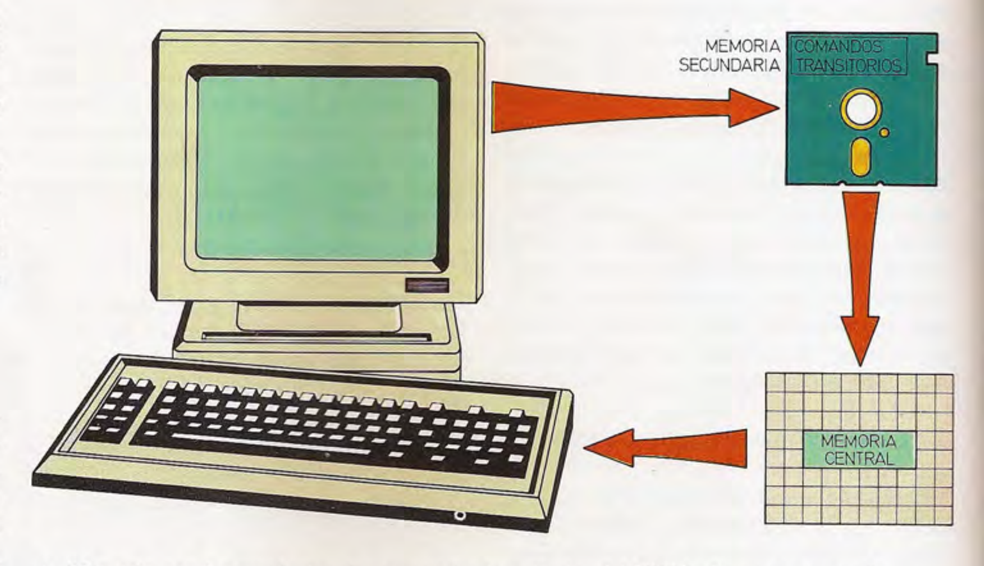
así disponible para reescribir en él un nuevo fichero.
El formato de la orden es:
A>*ERA x: Nombre del fichero. Tipo*
Una vez introducida esta orden en la máquina, se borrará el fichero cuyo nombre se especifica y que se encuentra almacenado en el disquete situado en la unidad *x*.
De nuevo es posible utilizar referencias ambiguas para especificar más de un fichero.
Es conveniente ejecutar una orden DIR antes de utilizar el comando ERA. El motivo no es otro que comprobar previamente la existencia de los ficheros que se desea eliminar. Asimismo, es conveniente utilizar el comando DIR a posteriori, para comprobar que el borrado se ha realizado de forma correcta.



Los comandos residentes o permanentes del CP/M se cargan en la memoria central de la máquina junto con el sistema operativo.

REN

REN se emplea para otorgar un nuevo nombre a un fichero. La forma que adopta esta orden es:
A>*REN x: Nuevo nombre. Tipo = Antiguo nombre. Tipo*
La referencia *x* identifica a la unidad en la que se encuentra el disquete que contiene el fichero. En el contexto de esta orden, no cabe utilización de referencias ambiguas, ya que no lo permite el sistema operativo CP/M.



A diferencia con los residentes, los comandos transitorios del CP/M sólo se cargan en la memoria central del ordenador cuando son invocados por el usuario desde el teclado.

SAVE

Se utiliza para trasladar a disco el contenido de la memoria central. La información que se transmite al disco se encuentra almacenada en el área de la memoria primaria destinada a los programas de usuario (TPA). El formato de esta orden es el que se indica a continuación:
A>*SAVE Y x: Nombre de fichero*
En esta ocasión, *x* representa la unidad en que se encuentra el disquete que recibirá los datos, mientras que Y representa el número de páginas de memoria que se desea almacenar. En el contexto del sistema operativo CP/M, la página es un conjunto de 256 caracteres. A la hora de utilizar el comando SAVE, hay que adoptar dos precauciones. La primera de ellas es reparar en que esta orden borra cualquier fichero cuyo nombre coincida con el especificado; de ahí que sea conveniente activar el comando DIR con anterioridad. La segunda, supone definir con exactitud el número de páginas de memoria a guardar; con ello se conseguirá ahorrar espacio de disco. Al respecto, cabe señalar que existen órdenes especiales que revelan la distribución de los datos almacenados en la zona TPA.

TYPE

El cometido de TYPE es visualizar ficheros que contengan caracteres imprimibles. Unicamente permite su visualización; la modificación de los datos contenidos en el fichero es tarea del editor del sistema operativo o de un programa de tratamiento de texto, dependiendo del tipo de datos que residan en el mismo. Su formato es:
A>*TYPE x: Nombre del fichero*
En la expresión, *x* identifica a la unidad en que se encuentra el disquete que aloja al fichero.

CONTROL-C

Constituye la orden de "arranque en caliente" del sistema operativo CP/M. Este comando se introduce pulsando simultáneamente las teclas <CTRL> y C.
El efecto que produce es el de restaurar la configuración de la memoria a un estado predefinido. Sus funciones principales son:

a. La interrupción de un programa en curso de ejecución y el regreso al nivel de órdenes propias del CP/M.
b. Catalogar un nuevo disquete cuando se introduce en una de las unidades de lectura.

CONTROL-E

Este comando se utiliza para indicar al ordenador que las órdenes que recibe en una línea continúan en la siguiente. Se introduce pulsando simultáneamente las teclas <CTRL> y E.

CONTROL-P

Su misión es activar o desactivar la impresora, dependiendo del estado de la misma. Cuando la impresora se activa, las salidas por pantalla también se reproducen en la impresora. Se introduce pulsando las teclas <CTRL> y P al unísono.

CONTROL-X

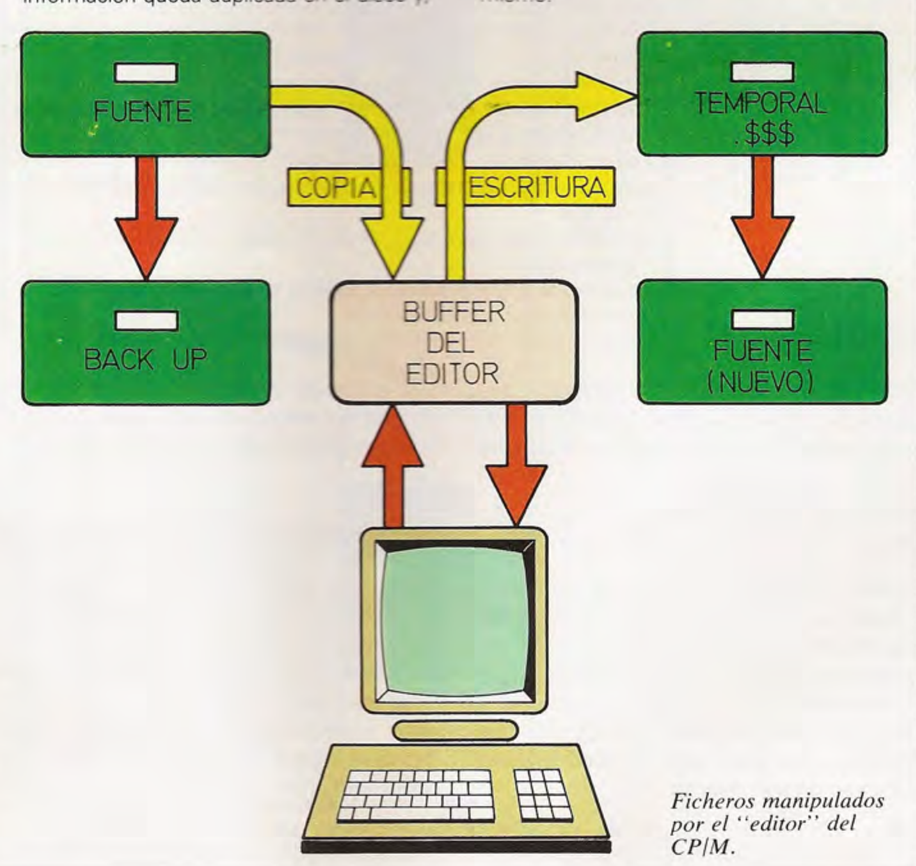
La misión de CONTROL-X es cancelar e impedir que el ordenador ejecute una serie de órdenes introducidas a través del teclado. Se emplea normalmente cuando el número de errores de mecanografía es elevado y, por lo tanto, es más práctico empezar de nuevo la introducción. En algunas versiones de CP/M-80, concretamente la 1.3, se sustituye la X por U a la hora de teclear la mencionada orden.

COMANDOS TRANSITORIOS

Los comandos transitorios se encuentran almacenados, habitualmente, en disco. Sólo se cargan en la memoria central y se ejecutan cuando son invocados específicamente. En la práctica, estos comandos se comportan como si fueran programas, ignorando completamente al CP/M; sólo regresan a ese nivel tras su ejecución. Algunos de estos comandos exigen una serie de parámetros complementarios. Estos se introducen a continuación del co-

El editor del sistema operativo CP/M

La escritura de un programa o la modificación de un fichero es una tarea cotidiana para el programador o para el usuario avezado. A través del teclado del ordenador, se van introduciendo las distintas sentencias de que consta el programa, o los datos que deben engrosar el fichero; sentencias y datos que son visualizados sobre la pantalla y modificados o corregidos por la acción de ciertas órdenes dadas a través del teclado.
El Editor es el programa que apoya al usuario en estos menesteres: toma los caracteres del teclado y los presenta sobre el monitor, permite corregir los errores que puedan cometerse durante el proceso de escritura (brindando órdenes que modifican, eliminan o aumentan la información que se ha tecleado)..., y se ocupa de transmitir esta información al disco o a la memoria secundaria del ordenador, una vez concluido el proceso de edición.
En el sistema operativo CP/M la gestión de las modificaciones y, en general, de la edición de un fichero, se apoya en una partición de la memoria denominada "buffer del editor". En esta partición se carga el fichero fuente u original a modificar (al que denominaremos, sencillamente, "fichero"), al tiempo que se genera en el disco un fichero vacío, de nombre: *fichero*. $$$.
Sobre la información contenida en este buffer es, precisamente, donde actúan los comandos activados a través del teclado. Una vez que se estima correcto el resultado, se dará la orden para almacenar el fichero. En este preciso instante ocurren dos procesos simultáneos. El primero es el trasvase de la información contenida en el buffer de memoria hacia "*fichero*. $$$", el cual cambia su nombre por el fichero fuente original, y se almacena como el nuevo fichero fuente. El segundo proceso coincide con el cambio de nombre del antiguo fichero fuente, que pasa a denominarse *fichero BAK* y a actuar como "back-up" (copia de seguridad) del otro fichero. A raíz de este último proceso, la información queda duplicada en el disco y, por lo tanto, resultará difícil su borrado o pérdida accidental.
Si el tamaño del fichero es superior al del buffer de memoria, el método que se sigue es idéntico, con la única salvedad de que el fichero es actualizado por etapas y el cambio en el nombre de los ficheros se realiza al actualizar la última porción del mismo.



Ficheros manipulados por el "editor" del CP/M.

APLICACIONES DEL COMANDO STAT

Comando	Significado
STAT x:	Indica el espacio libre en el disco del dispositivo x.
STAT x: Nombre fichero	Indica el espacio ocupado en el disco por el fichero.
STAT DEV:	Hace visible la asignación actual entre dispositivos físicos y lógicos.
STAT VAL:	Visualiza las posibles asignaciones de dispositivos físicos y lógicos.
STAT x: DSK	Visualiza la información referente a cómo se almacenan los datos en el disco del dispositivo x.
STAT x: R/O	Asigna al disco del dispositivo x una protección de forma que sólo pueda leerse información.
STAT log:=phy	Asigna el dispositivo físico phy al dispositivo lógico log.

SUBCOMANDOS DEL EDITOR (ED)

Comando	Significado
#A	Añade líneas del fichero original al buffer de editor.
E	Fin de la sesión de edición.
H	Localiza el principio del fichero que se está editando.
O	Borra el fichero editado.
Q	Fin de la sesión de edición. Los ficheros no se modifican.
nK	Elimina n líneas del fichero.
nL	Avanza n líneas el cursor.
U	Transforma minúsculas en mayúsculas.
V	Visualiza números de línea.

PARAMETROS BASICOS DEL COMANDO PIP

Parámetros	Significado
D#	Suprime todos los caracteres que sobrepasen en la transferencia de información la columna #–ésima.
E	Eco en la consola al realizar una copia.
G#	Permite la copia de ficheros desde áreas de usuarios distintas a la usada.
U	Convierte letras minúsculas a mayúsculas en la transferencia de información.
N	Almacena añadiendo números de línea a cada línea.
V	Verifica que la copia es correcta.



Los comandos del CP/M permiten organizar y mantener en orden los archivos de información almacenados en el ordenador.



El editor engloba a las herramientas que brinda el sistema operativo CP/M para la escritura y modificación de los datos memorizados en los ficheros.

mando y antes de pulsar la tecla de retroceso de carro (RETURN o ENTER). El formato general de esta orden es el que se indica a continuación.

A>*Comando Parámetros*

STAT

Pueden suministrar información acerca de ficheros o grupos de ficheros, o bien información relativa a dispositivos físicos y lógicos del CP/M. Cabe precisar que los dispositivos lógicos coinciden con funciones como, por ejemplo: introducción de órdenes, recepción de información, envío y listado de información...

PIP

El comando PIP es un comando destinado a la copia de información de una zona a otra de la memoria central, entre dispositivos periféricos o de un fichero a otro.

ED

ED es el comando adecuado para llamar al editor del sistema operativo CP/M. Por medio del editor es posible modificar y crear ficheros de datos y programas. Una vez invocado, el editor opera gobernado de una serie de parámetros que permiten funciones tales como eliminar o añadir líneas y caracteres. Algunos de los referidos parámetros aparecen definidos en el cuadro adjunto.

SUBMIT

La tarea del comando SUBMIT es instruir al ordenador para que ejecute una serie de órdenes del sistema operativo CP/M, sin que exista intervención por parte del operador. Para ello se crea un fichero, de nombre *"Nombre del fichero. SUB"*, por medio del editor del sistema operativo. En este fichero se escriben los diferentes comandos —uno por línea—, que han de ser ejecutados. Una vez cerrado el fichero, y para que se ejecute el conjunto de órdenes que contiene, es preciso teclear:

A>*SUBMIT Nombre del fichero. SUB*

La utilidad de este comando reside en la interesante posibilidad de ejecutar conjuntos de órdenes de tipo repetitivo, con comodidad y sin exigir la atención parcial del usuario.

Wordstar (2)

El complemento MAILMERGE. Instalación y prueba del tratamiento de textos

Tras la definición y el estudio de los principales comandos, acometido en el primer capítulo dedicado al WORDSTAR, llega el momento de analizar las opciones complementarias y la forma de instalación y prueba inicial de este tratamiento de textos.

El complemento básico del WORDSTAR lo aporta el paquete MAILMERGE, orientado a la producción de cartas y otros documentos especiales.

MAILMERGE

Este es el nombre que recibe un paquete de aplicación, complementario del WORDSTAR y concebido como opción del mismo, cuya especialidad reside en la edición de cartas en serie ("mailings"). Puede adquirirse junto al WORDSTAR, o bien separado del resto de la aplicación. Cabe señalar que la ejecución de este paquete complementario impide simultanear la impresión de un documento y la edición de otro texto distinto. En consecuencia, al ejecutar el comando MAILMERGE se entrará en estado de impresión, y hasta que ésta concluya, el procesador no podrá dedicarse a ninguna otra función.

Por supuesto, las posibilidades del MAILMERGE incluyen por completo todas las propiedades de impresión del WORDSTAR, descritas en el capítulo anterior. Si bien, ofrece además una serie de opciones adicionales entre las que cabe señalar las siguientes:

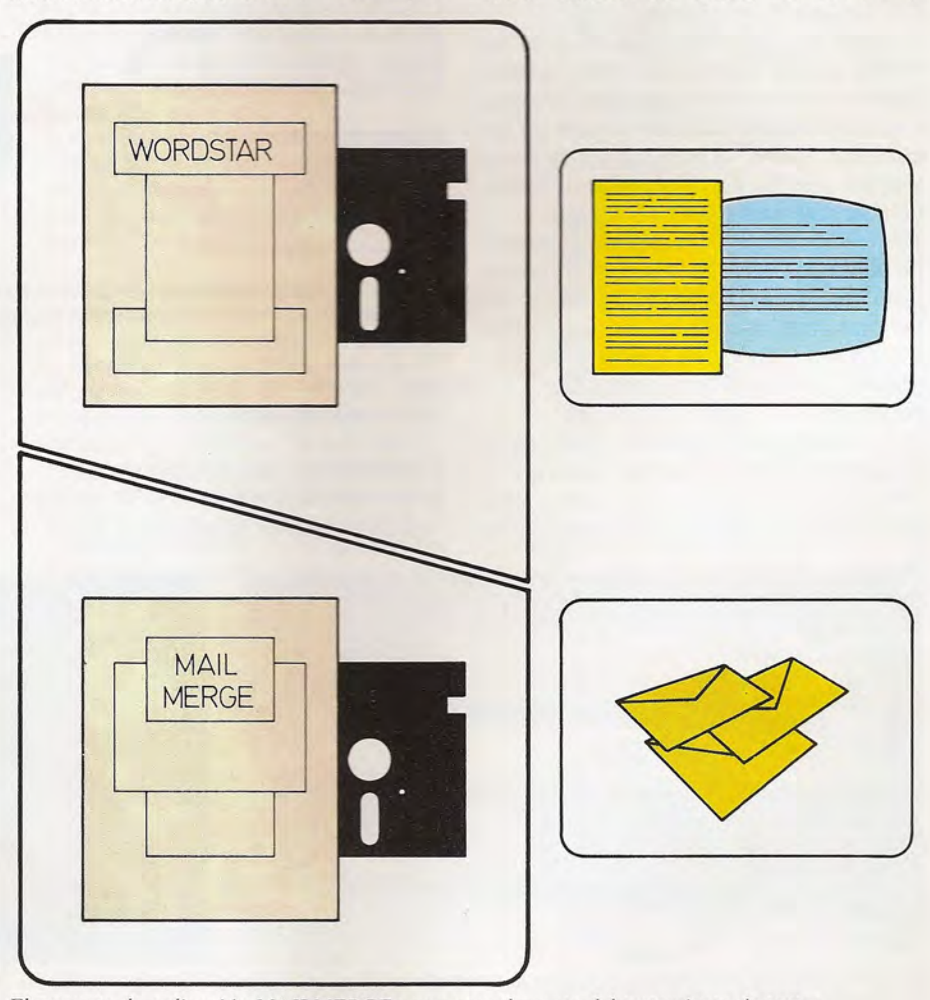
1. Ficheros de datos

Se utilizan para almacenar listas o cualquier otra información que debe insertarse en documentos mediante el MAILMERGE. Para crear y/o modificar estos ficheros se puede utilizar la opción normal de edición del WORDSTAR, aunque también puede emplearse un programa adicional, de la firma MICROPRO, encargado de capturar datos: el DATASTAR.

Una tercera posibilidad para gestionar los ficheros de datos, consiste en realizar programas de usuario adecuados para realizar el oportuno tratamiento de los mismos.

2. Toma externa de datos

Esta opción sirve para insertar información variable en un documento, e imprimir un ejemplar del mismo con los datos tomados del exterior. Un típico ejemplo de toma externa de datos, realizable me-

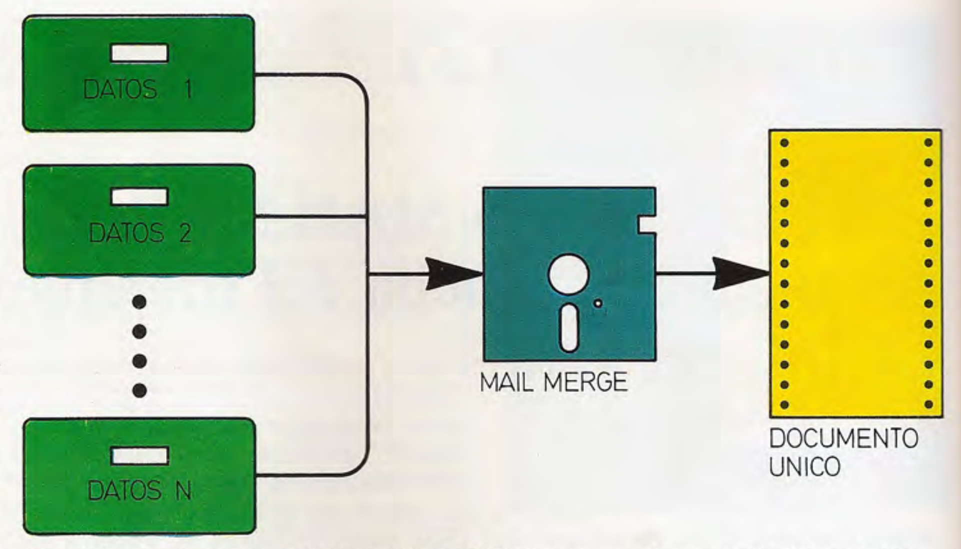


El paquete de aplicación MAILMERGE es un complemento del tratamiento de textos WORDSTAR. Su presencia permite realizar determinadas funciones especiales muy útiles, particularmente, a la hora de producir cartas en serie (mailings).

diante el MAILMERGE, es la producción de cartas personalizadas. A partir de un texto básico, a incluir en todas las cartas, se sustituirán determinados "huecos" por datos externos, por ejemplo: el nombre y la dirección de la persona a quien se dirija la carta.

La información a insertar puede obtenerse de un fichero de datos como el señalado anteriormente. Sin embargo, también puede ser introducida directamente por el operador a través del teclado, e incluso pueden determinarse valores al principio del documento para que se inserten en varios sitios del mismo.

Para controlar la fusión de los datos externos con el texto básico de la carta, se utilizarán controles embebidos en dicho texto. En todo caso, tanto la edición del texto básico como la de los controles, se efectuará a través de las funciones de edición del WORDSTAR.



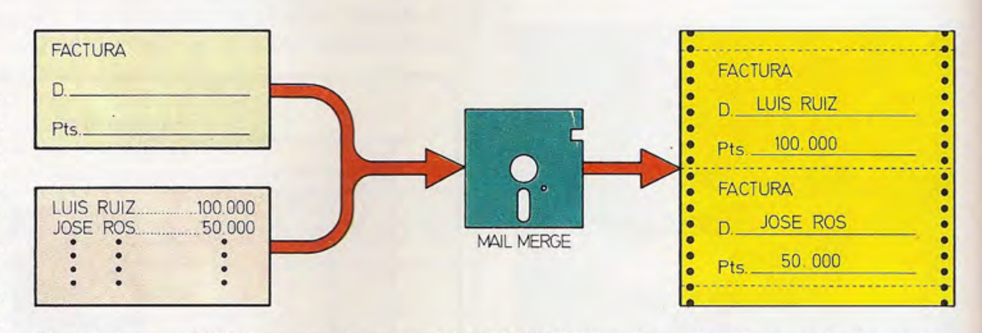
Una de las funciones primordiales del MAILMERGE consiste en la conjunción del contenido de varios archivos de información en un único documento escrito.

3. Impresión en cadena

Cuando un documento es de gran tamaño, puede resultar incómodo editarlo desde un único archivo de datos. En este caso, el MAILMERGE permite que un documento "llame" a otro archivo de documentos por su nombre; de esta forma, será posible imprimirlos en cadena.

Tal posibilidad también resulta útil cuando se utiliza una frase o párrafo de forma muy repetitiva en uno o más documentos; estas porciones de texto se pueden guardar en archivos separados y pueden ser "llamadas" por tantos documentos como sea necesario.

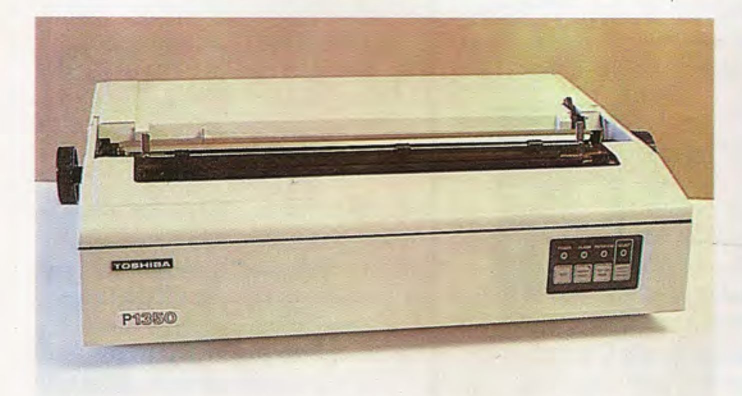
Otro método para gestionar la impresión en cadena de varios textos, pasa por la creación de un archivo de control; éste almacenará los documentos adecuados para imprimir las distintas partes de un mismo documento.



Una de las posibilidades más relevantes del MAILMERGE es la producción en serie de cartas; "cruzando" un archivo de datos con el archivo que almacena el texto común.

4. Obtención de múltiples copias

El WORDSTAR dispone de varios métodos para obtener varias copias de un documento. Uno de ellos consiste en la inclusión de comandos destinados a la producción de varias copias repetidas; si se desea que el número de copias sea variable, puede especificarse esta circunstancia a través de comandos al efecto. Por lo



La impresora es el periférico que plasma la tarea desarrollada por el usuario con la colaboración del paquete para el tratamiento de textos. Esta puede ser de diverso tipo y capaz de obtener copias de mayor o menor calidad.



Para el tratamiento de textos, es conveniente que el ordenador esté complementado por una doble unidad de disco que permita obtener cómodamente copias de seguridad de los archivos.

demás, el usuario de MAILMERGE puede decidir la repetición de un documento tantas veces como estime oportuno.

5. Comunicaciones con el operador
Esta facilidad permite que, durante la impresión de un documento, el WORDSTAR pueda escribir mensajes destinados al operador: pidiéndole que complete cierta información, o que active un tipo especial de papel continuo. La comunicación entre procesador y operador se establece mediante comandos especializados, insertados en el archivo que contenga el texto.

6. Formateo durante la impresión
El formateo estándar se realiza mediante el WORDSTAR durante la ecritura; no obstante, se puede utilizar la opción MAILMERGE, de tal forma que quede abierta la posibilidad de insertar información variable en determinadas posiciones del texto.

INSTALACION DEL WORDSTAR

Para que el WORDSTAR pueda funcionar en un ordenador personal, hay que considerar ciertos requisitos mínimos en cuanto a instalación. Básicamente, son necesarios los siguientes elementos: un ordenador personal con suficiente memoria RAM y equipado con el sistema operativo adecuado (CP/M o MS/DOS), una unidad para disco flexible (o unidad de disco rígido), un terminal (teclado y pantalla), una impresora y, por supuesto, la aplicación para el tratamiento de textos WORDSTAR.

Contenido del paquete WORDSTAR
Junto con el paquete de aplicación WORDSTAR, se recibe una documentación para su manejo y un disco flexible que incluye los siguientes archivos:
—WSU-COM. Contiene información necesaria para proceder a la instalación del WORDSTAR.
—MAILMERGE-OVR. Contiene información necesaria para proceder a la instalación del MAILMERGE.
—INSTALL-COM. Programa de instalación que utilizará el archivo WSU-COM y MAILMERGE-OVR.

Impresoras para procesos de textos

Uno de los elementos hardware más importantes para realizar el tratamiento de textos es la impresora; el periférico encargado de obtener la copia escrita de los textos procesados. De ella dependerá la calidad final de los documentos producidos; de poco servirá disponer de una aplicación potente y ágil, si el periférico en que finalmente se imprime el resultado no es satisfactorio.
Dentro de la enorme cantidad de impresoras disponibles en el mercado informático, cabe distinguir los siguientes tipos básicos

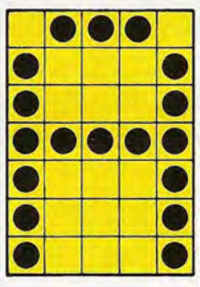
Impresoras de bola
Trabajan con una bola —de ahí su nombre— que contiene todos los caracteres que pueden ser escritos. La impresión en este tipo de periférico se realiza carácter a carácter y, por lo tanto, su velocidad suele ser relativamente baja. A cambio de esta lentitud, ofrece una escritura de alta calidad.
El mecanismo para la escritura de cada carácter, se fundamenta en el giro de la bola, de tal forma que el carácter a escribir quede en la posición de escritura enfrentado al papel.

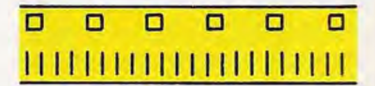
Impresoras de matriz de puntos
En este caso, la escritura de cada carácter corre a cargo de una matriz de agujas. Activando y desactivando los distintos puntos de la matriz se obtendrá la impresión de carácter que se desee.
Dentro de las impresoras de matriz de puntos, existen dos tipos fundamentales; ambos permiten obtener informes escritos de inferior calidad a la que se logra con impresoras de bola. El primer tipo se basa en la escritura carácter a carácter, mientras que las impresoras del segundo tipo escriben simultáneamente todos los caracteres de una línea. En todo caso, la velocidad de impresión es bastante más elevada que la disponible en las impresoras de bola.

Impresoras de banda
La impresión se realiza por medio de una banda de acero que contiene todos los caracteres generales. Esta gira continuamente a gran velocidad de forma que, en el momento oportuno, un pequeño golpe producido por un martillo permite la impresión del carácter en el papel. El martillo empuja a la banda de acero contra una cinta entintada que se encuentra entre la banda y el papel.

Impresoras de margarita
Junto con las de bola, las impresoras de margarita son las que producen documentos finales de más alta calidad. El dispositivo de escritura es una rueda, denominada margarita, cuyos brazos contienen los distintos caracteres. Para imprimir uno de ellos, la margarita se posiciona de tal forma que, mediante un pequeño impulso de un martillo, el brazo con el carácter adecuado golpea la cinta entintada contra el papel.

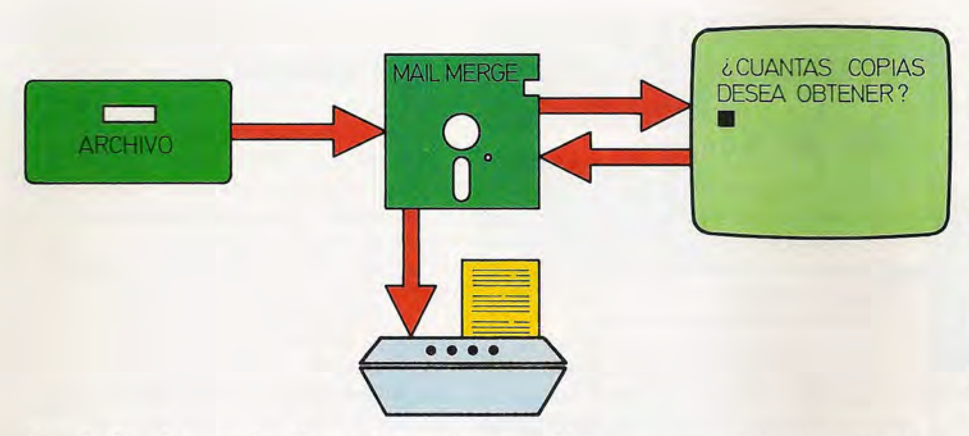




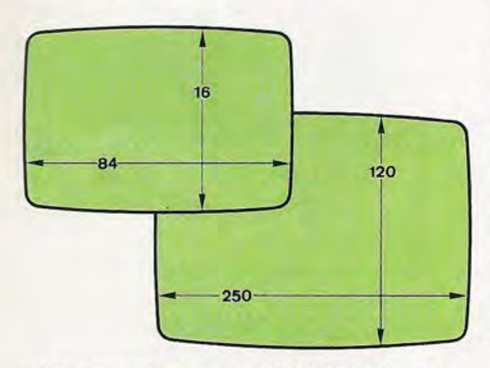




Mecanismos básicos utilizados por las impresoras de bola, matriz de puntos, banda y margarita.



Para facilitar la comunicación con el operador, es posible insertar comandos en el archivo. De esta forma, antes de proceder a su escritura, el ordenador pedirá al usuario que matice la tarea a realizar.



El tratamiento de textos WORDSTAR puede trabajar con pantallas de un mínimo de 16 líneas por 80 columnas y un máximo de 120 líneas por 250 columnas.

—WSMSGS.OVR. Mensajes del WORDSTAR.
—WSOVLY 1.OVR. Mensajes del WORDSTAR.
Dado que el "paquete" MAILMERGE es opcional, el archivo MAILMERGE-OVR sólo estará incluido si, en efecto, se adquiere junto al WORDSTAR. En el caso de que la compra se haga por separado, se recibirá un nuevo disquete con el archivo correspondiente.

Características del ordenador personal
Un primer condicionante básico del ordenador personal que se utilice para "soportar" al WORDSTAR, es el tamaño de su memoria RAM; por supuesto, cabe dar por descontado que su sistema operativo es capaz de "aceptar" la aplicación.
El volumen mínimo de la memoria de lectura/escritura es de 48 Kbytes, incluyendo espacio para el sistema operativo y áreas para el almacenamiento de texto.
Si se desea simultanear la impresión de texto con la edición de otros archivos, es necesario disponer de 3 Kbytes adicionales. En cualquier caso, si se pretende sacar un buen partido de todas las posibilidades que ofrece el WORDSTAR, es recomendable disponer de más espacio de memoria principal del estrictamente necesario.
Es importante no confundir el concepto memoria RAM (memoria principal del ordenador que permite la lectura y escritura de información) con memoria secundaria o de masa (memoria auxiliar externa, habitualmente a base de discos flexibles).

Características del terminal
Puede utilizarse cualquier tipo de terminal, siempre que la pantalla tenga un mínimo de 16 líneas y 84 columnas y un máximo de 120 líneas por 250 columnas. También es necesario que utilice el código ASCII y sea capaz de posicionar el cursor en cualquier punto de la pantalla. Como características adicionales conviene que permita la inserción y borrado de líneas, y que posea atributos de campos, tales como inversión video, parpadeo, realce de intensidad, etc.
En el caso más frecuente, el teclado y la pantalla forman parte de la configuración básica del propio ordenador personal. No obstante, ciertos equipos constan exclusivamente de la unidad central y exigen la adopción de un terminal independiente.
Para que el WORDSTAR pueda funcionar con el terminal elegido, hay que especificar las características de dicho terminal respondiendo a un menú durante el proceso de instalación.

Características de la impresora
El WORDSTAR puede utilizar prácticamente cualquier tipo de impresora de las existentes en el mercado. Por supuesto, las funciones de la impresora van a determinar la viabilidad o inviabilidad de algunas de las opciones de impresión del procesador de textos. Por ejemplo, para escribir un documento con letra negrilla y subrayado, es imprescindible que la impresora acoja ambas posibilidades.

Características de las unidades de disco flexible
Para la cómoda y eficaz manipulación de los archivos, es imprescindible contar con una unidad de disco simple o doble (con capacidad para uno o dos disquetes). En el primer caso, si sólo se dispone de una unidad, será posible trabajar correctamente, si bien, no quedará abierta la opción de realizar copias de seguridad que garanticen la conservación de los archivos en el caso de que se estropee un disco de trabajo. Con una doble unidad de disco queda solventado tal inconveniente. El espacio recomendable para el almacenamiento de archivos es de 70 Kbytes para ficheros especiales del WORDSTAR, más el adecuado para el almacenamiento de los archivos a editar.

PRUEBA DEL WORDSTAR

Cuando se activa el WORDSTAR, ya sea de forma automática al finalizar su instalación, o bien al invocarlo escribiendo WS, debe borrarse el contenido de la pantalla de forma automática y aparecer en ella el mensaje inicial. En este último figurará el número de la versión y el número de serie del paquete de aplicación. Después de algunos segundos, o tras accionar cualquier tecla, debe aparecer el denominado menú sin archivo. En este punto conviene realizar algunas pruebas básicas:
—El posicionamiento del cursor.
—El archivo de mensajes.
—La función de borrado.
—La función de superposición.
—La función de edición.
—El funcionamiento de la impresora.
Una vez finalizada la comprobación de estas características, puede ya dar comienzo la sesión de trabajo con uno de los procesadores de textos más potente y completos que existen en el mercado.

Programando bucles

Estructuras cíclicas y decisiones de múltiple alternativa

A la hora de adentrarse en los secretos de la programación hay que disponerse a franquear con éxito el estudio de las estructuras de control. Este es uno de los aspectos más importantes de la programación en lenguaje BASIC.

Las estructuras de control llevan asociadas una serie de instrucciones cuya función primordial es la ruptura de la secuencia de ejecución normal del programa. Estas instrucciones permiten crear programas capaces de tomar decisiones en base a criterios establecidos por el programador. El resultado de la decisión puede llevar a ejecutar o no una determinada zona del programa, o bien puede controlar la ejecución repetitiva de una rutina un determinado número de veces.

Por el momento, las instrucciones BASIC de este tipo sometidas a examen se reducen a dos: GOTO (salto incondicional) e IF/THEN/ELSE (bifurcación condicional con toma de decisión). Este repertorio de instrucciones de control se completa en el presente capítulo con dos nuevas incorporaciones: FOR/NEXT y ON/GOTO.

PROGRAMACION DE BUCLES

El objetivo de un programa BASIC puede ser, sencillamente, calcular el importe total de tres artículos de distinto precio unitario y adquiridos en distinta cantidad. La forma de escribir o codificar un programa semejante, empleando el lenguaje BASIC, no plantea excesivos problemas. Una posible solución es la siguiente:

```
10 LET T=0
20 INPUT "CANTIDAD DEL ARTICULO"; C
30 INPUT "PRECIO DEL ARTICULO"; P
40 LET N=C*P
50 PRINT "CANTIDAD A PAGAR:"; N
60 LET T=T+N
70 INPUT "CANTIDAD DEL ARTICULO"; C
80 INPUT "PRECIO DEL ARTICULO"; P
90 LET N=C*P
100 PRINT "CANTIDAD A PAGAR:"; N
110 LET T=T+N
120 INPUT "CANTIDAD DEL ARTICULO"; C
130 INPUT "PRECIO DEL ARTICULO"; P
140 LET N=C*P
150 PRINT "CANTIDAD A PAGAR:"; N
160 LET T=T+N
170 PRINT "EL TOTAL A PAGAR ES:";T;
    "PESETAS"
180 END
```

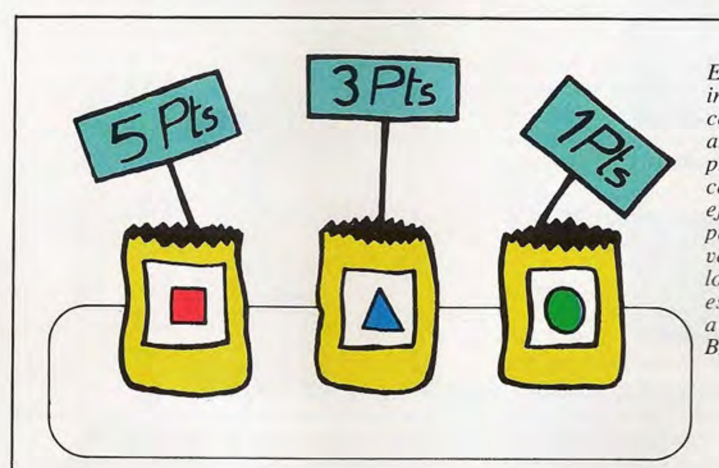
El programa resulta un tanto extenso, aunque, sin lugar a dudas, realiza la función encomendada.

La primera instrucción (10 LET T=0) pone a cero la variable que irá totalizando los importes o cantidades a pagar por los diversos artículos adquiridos. La zona delimitada por las instrucciones 20 a la 60 solicita la introducción de la cantidad (C) y precio (P) del primero de los artículos; tras ello, calcula el importe total del mismo (40 LET N=C*P) y lo almacena en la variable T (60 LET T=T+N).

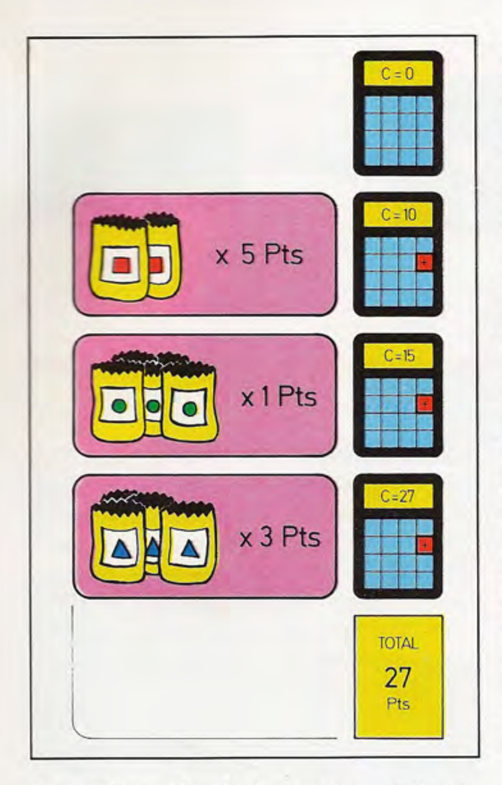
Como quiera que son tres los artículos que se adquieren, el programa repite la mencionada zona (instrucciones 20 a la 60) dos veces más (líneas 70 a la 110 y 120 a la 160). Por último ya sólo queda por presentar en la pantalla el importe total de la compra, memorizado en la variable T; de ello se encarga la instrucción 170.

Veamos un ejemplo de ejecución:

```
RUN
CANTIDAD DEL ARTICULO? 2
PRECIO DEL ARTICULO? 5
CANTIDAD A PAGAR: 10
CANTIDAD DEL ARTICULO? 5
PRECIO DEL ARTICULO? 1
CANTIDAD A PAGAR: 5
CANTIDAD DEL ARTICULO? 4
PRECIO DEL ARTICULO? 3
CANTIDAD A PAGAR: 12
EL TOTAL A PAGAR ES: 27 PESETAS
```



El simple cálculo del importe total de una compra de tres artículos, de distinto precio unitario, constituye un ejemplo adecuado para estudiar las ventajas que aportan los bucles o estructuras cíclicas a la programación BASIC.



El método más directo consiste en calcular por separado el importe total de cada artículo y totalizar la suma final. El programa al efecto no incorpora bucle alguno, sino que repite en tres ocasiones la misma operación antes de evaluar la suma final.



La moderación de la compra realizada hace que el programa no plantee ninguna dificultad. ¿Pero qué sucede cuando el número de productos es mayor? Suponga que se realiza una compra de más de cien productos o artículos distintos. Es evidente que la estructura del programa anterior no resultará adecuada. ¿Existe algún método que permita confeccionar el programa adecuado sin recurrir a la tediosa tarea de escribir una y otra vez idéntico bloque de operaciones?

Desde luego que hay soluciones. Este es precisamente uno de los cometidos fundamentales de las estructuras de control. Su puesta en práctica permite crear bucles en el programa cuya ejecución puede repetirse tantas veces como sea preciso. Una posible alternativa, bastante más racional, es la que propone el siguiente programa:

```
10 LET T=0
20 LET I=0
30 INPUT "CANTIDAD DEL ARTICULO"; C
40 INPUT "PRECIO DEL ARTICULO"; P
50 LET N=C*P
60 PRINT "CANTIDAD A PAGAR"; N
70 T=T=N
80 LET I=I+1
90 IF I<3 THEN GOTO 30
100 PRINT "LA CANTIDAD TOTAL A PAGAR
    ES:"; "PESETAS"
110 END
```

A todas luces, el nuevo programa mejora al anterior. En primer lugar ahorra tiempo al programador a la hora de confeccionarlo y, por otra parte, el programa resulta bastante más versátil. Con un mínimo esfuerzo es posible acondicionar el programa para que sea capaz de totalizar la adquisición de cualquier número de productos distintos. Para ello, sólo hay que definir cuántas veces debe repetirse el bucle de cálculo.

El número de veces que hay que repetir la rutina comprendida entre las líneas 30 y 70 es controlable actuando en distintos puntos del programa: modificando el valor inicial de I (línea 20), alterando su valor final (línea 90), o incluso variando el incremento que experimenta la variable I en cada nueva ejecución de la rutina (línea 80).

Este ha sido un ejemplo ilustrativo de la ventaja que supone el empleo de instrucciones de control a la hora de realizar tareas repetitivas. No obstante, las posibilidades de esta técnica no se reducen a lo expuesto. El ejemplo siguiente revela con mayor elocuencia las ventajas que aporta este método, aplicándolo a un programa capaz de obtener una lista de los números pares, desde el 0 al 100:

```
10 LET C=0
20 PRINT I*2
30 I=I+1
40 IF I<50 THEN GOTO 20
50 END
■
```

En esta ocasión, la variable que se va incrementando, y que a partir de ahora llamaremos variable *contadora,* interviene directamente en los cálculos. Esta posibilidad ofrece ventajas muy interesantes en determinados casos.

TOTAL A PAGAR
T: 0 Pts
CONTADOR DE CICLOS
I: 0
BUCLE DE CALCULO
N=C•P
CALCULO DEL IMPORTE POR CADA ARTICULO
T=T+N
TOTALIZAR IMPORTES SUCESIVOS
HAY MAS ARTICULOS (I<3)
SI
NO
TOTAL: 27pts

Con la ayuda de la instrucción IF/THEN, es posible reducir la longitud del programa. El mismo bucle de cálculo se ejecutará en tres ocasiones, una para cada artículo. Al salir del ciclo, el programa sumará los importes parciales.

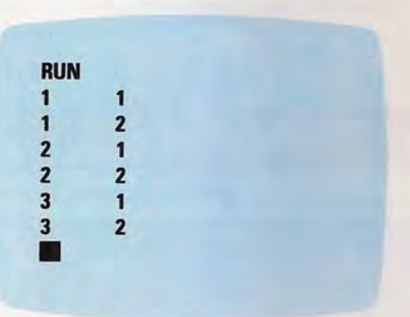
LA ESTRUCTURA FOR/NEXT

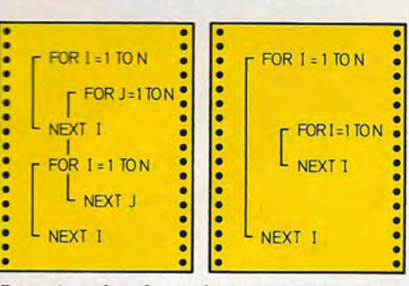
Se observa pues que las tareas rutinarias pueden ejecutarse cómodamente en el seno de *bucles*. Estos consisten, sencillamente, en un bloque de instrucciones que se ejecutan una y otra vez, hasta que se cumple una determinada condición.
La estructura más sencilla para la ejecución de bucles, y a su vez la más utilizada, es la FOR/NEXT. Esta queda definida en la figura adjunta, en la que aparecen un diagrama de flujo representativo de su actuación.
La instrucción o instrucciones que componen la referida estructura FOR/NEXT deben contemplar cuatro elementos esenciales:
— Delimitación de los extremos del bucle; hay que precisar el principio y final del mismo.
— Indicación de los valores inicial y final de la variable que se utilizará como contador.
— Definición del incremento o, de forma más general, de la variación que ha de experimentar la variable contadora con cada nueva ejecución del bucle.
— Indicación de la variable que va a utilizarse como contador.
En el BASIC, todo ello queda plasmado dentro de una estructura formada por dos instrucciones: FOR y NEXT.
En el argumento de la instrucción FOR hay que especificar cuál será la variable contadora, indicar el valor inicial y final de la misma (TO) y, además, precisar el incremento que experimentará dicha variable (STEP) en cada nueva ejecución del bucle.
Hay que tener en cuenta que la variación que debe afectar a la variable contadora al completar cada ciclo puede ser incluso negativa.
El formato general de la instrucción FOR es el siguiente:

Bucles anidados

Los bucles anidados constituyen una técnica de programación que consiste en introducir bucles FOR/NEXT dentro de otros bucles de la misma naturaleza. Este es un método muy frecuentemente utilizado en las tareas de programación.
Su puesta en práctica exige adoptar una precaución básica: los bucles deben "anidarse" de tal forma que cada uno de ellos esté completamente incluido dentro del otro. A su vez, dos bucles, uno anidado dentro del otro, no pueden compartir la misma variable contadora, sino que ésta ha de ser distinta.
El programa adjunto ilustra la actuación de un bucle anidado.
Las instrucciones 20, 30 y 40 constituyen un bucle FOR/NEXT anidado dentro de otro bucle exterior, este último delimitado por las líneas 10 y 50. El bucle externo utiliza la variable I como contador, mientras que el bucle anidado o interior emplea al respecto la variable J.
La instrucción 30 (PRINT I, J) permite observar cómo discurre la ejecución del programa. Para cada valor de la variable contadora I (bucle exterior), se ejecutan todos los posibles ciclos del bucle anidado (para J=1 y para J=2). Ello se debe a que el retorno del bucle externo (NEXT I) sólo se produce una vez que se ha salido del bucle interior y la ejecución rebasa la instrucción 40.

```
10 FOR I=1 TO 3
20 FOR J=1 TO 2
30 PRINT I,J
40 NEXT J
50 NEXT I
60 END
```



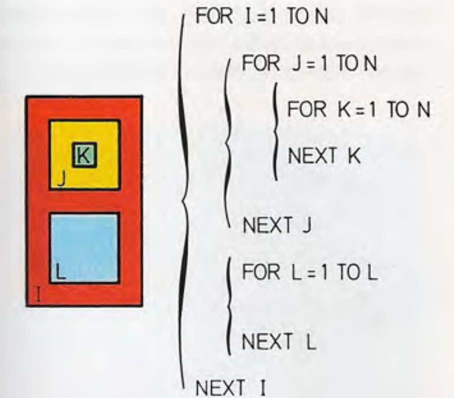


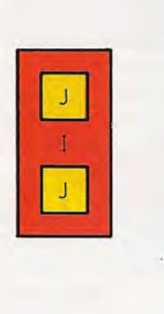
Dos ejemplos de anidamiento incorrecto de bucles FOR/NEXT.

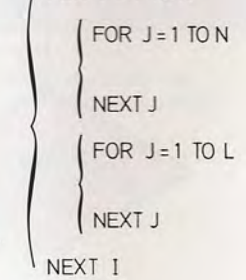
Algunos dialectos BASIC permiten cerrar los bucles anidados por medio de una sola instrucción NEXT, conteniendo ésta a todas las variables contadoras, ordenadas y separadas por comas.
En el ejemplo anterior, la instrucción de la línea 40 quedaría, en tal caso, de la siguiente forma:

40 NEXT J,I

Hay que tener en cuenta, no obstante, que esta posibilidad no es compartida por todos los dialectos BASIC; de ahí que, en principio, sea oportuno recurrir al método más general de cerrar cada bucle con una instrucción NEXT específica.







Ejemplos de anidamiento correcto de estructuras FOR/NEXT.

FOR/NEXT

Ejecuta el bucle encerrado entre ambas instrucciones tantas veces como sea necesario: hasta que la variable contadora alcance el valor de la expresión 2, partiendo el valor inicial establecido por la expresión 1 y en incrementos sucesivos dados por la expresión 3.

Formato: FOR <variable>=<expr.1> TO <expr.2> STEP <expr.3>

.

NEXT <variable>

Ejemplos: FOR I=1 TO 5 STEP 1.5

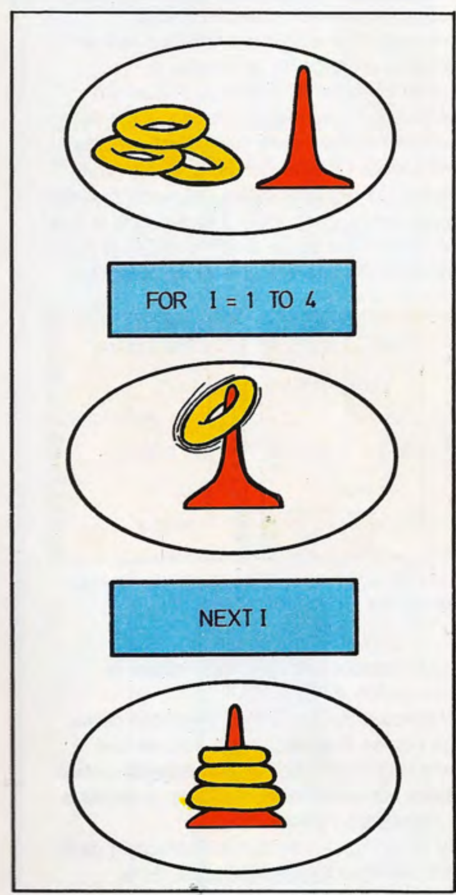
.

NEXT I

FOR K=A TO B STEP −Z

.

NEXT K



La estructura de control FOR/NEXT facilita la programación de tareas repetitivas. En la primera línea de la mencionada estructura (FOR/TO/STEP) se definen los valores inicial y final, además del incremento, de la variable que contabilizará los recorridos a través del bucle. A su vez, la instrucción NEXT pondrá fin al conjunto de instrucciones que constituyen el bucle definido, devolviendo el control a la línea FOR/TO/STEP.

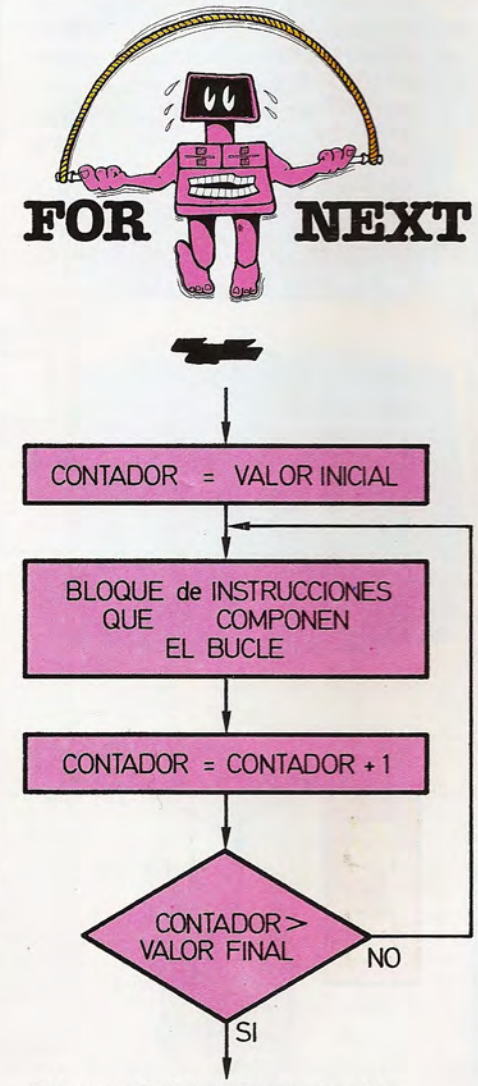


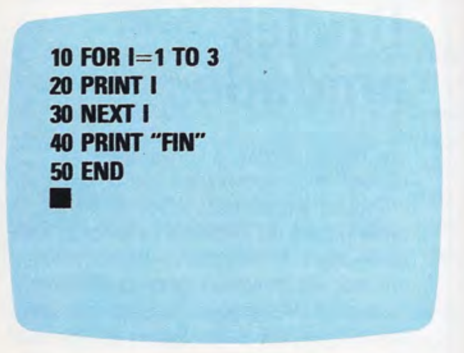
Diagrama de flujo representativo del funcionamiento de una estructura de bucle del tipo FOR/NEXT.

FOR <variable>=<valor inicial>TO <valor final> [STEP <incremento>]

De todos los elementos citados al principio, sólo queda por concretar el que se refiere a la delimitación del bucle. Ello se consigue fácilmente colocando una instrucción NEXT al final del bloque de instrucciones que constituyen el bucle. Su formato es tan simple como el que sigue:

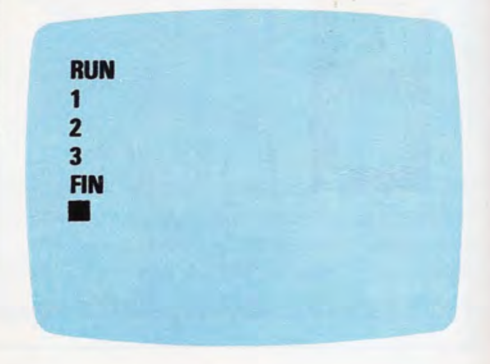
NEXT <variable>

Por supuesto, la variable ha de coincidir con la utilizada como variable contadora en el argumento de la correspondiente instrucción FOR.
Veamos un sencillo ejemplo:



```
10 FOR I=1 TO 3
20 PRINT I
30 NEXT I
40 PRINT "FIN"
50 END
■
```

Al ejecutar el programa se observa que el bucle, constituido en este caso por una única instrucción (20 PRINT I), es ejecutado tres veces.
Tras recibir la orden RUN, se ejecuta por primera vez la instrucción 10. Esta dicta al ordenador que debe repetir la ejecución del bucle mientras que el valor de la variable contadora I (inicializada a 1) no exceda del límite 3. En este caso, al omitir la indicación del incremento sucesivo que debe afectar a I (zona STEP), la máquina interpretará que debe incrementarla en una unidad tras cada recorrido del bucle.



```
RUN
1
2
3
FIN
■
```

En definitiva, el bucle comenzará a ejecutarse por primera vez para un valor de la variable contadora igual al valor inicial (I=1). Tras ejecutar la instrucción 20, se llegará a la que pone fin al bucle: NEXT I. En ese preciso instante, la variable contadora se incrementará en una unidad y el ordenador regresará a la instrucción FOR (instrucción 10). En ella, comprobará el

Simulación de la estructura ON/GOTO

La disparidad de los dialectos del lenguaje BASIC que equipan los distintos ordenadores se traduce en muy variados inconvenientes. Uno de ellos es la posible ausencia de la instrucción ON/GOTO, lo que, en principio, parece obligar al programador a olvidarse de esta eficaz herramienta para la toma de decisiones de múltiple alternativa. Aun cuando éste sea el caso, no hay por qué renunciar a esta útil estructura de control, ya que es

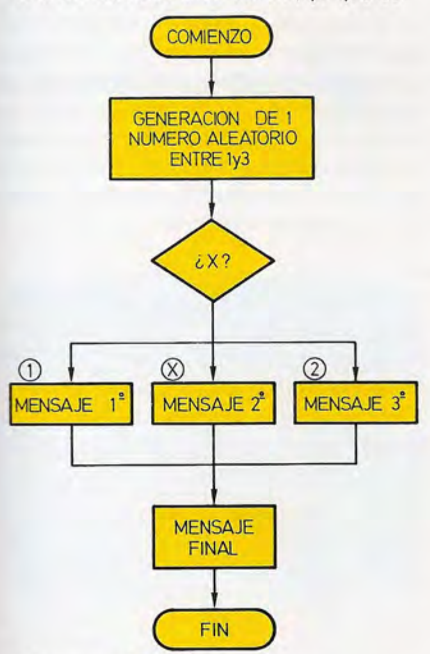


Diagrama de flujo del programa "Pronósticos", construido en base a una estructura ON/GOTO.

posible simularla por medio de otras instrucciones BASIC más habituales. La descripción va a partir del siguiente ejemplo:

```
100 X=(RND*3)+1
110 ON X GOTO 200, 300, 400
200 PRINT "PRONOSTICO: 1"
210 GOTO 600
300 PRINT "PRONOSTICO: X"
310 GOTO 600
400 PRINT "PRONOSTICO: 2"
600 "¡SUERTE Y A POR LOS CATORCE!"
700 END
```

Se trata de una zona de programa BASIC que incluye una toma de decisión múltiple, y cuyo correspondiente diagrama de flujo aparece en la figura adjunta. El cometido de este simple programa es la emisión de pronósticos para rellenar quinielas al azar.
La instrucción 100 incluye la función RND: una herramienta del vocabulario BASIC adecuada para la generación de números aleatorios y cuyo estudio se acometerá en un próximo capítulo de la obra. Por el momento, basta con saber que al ejecutar la referida instrucción, la variable X adoptará un valor comprendido entre 1 y 3. Este valor será el que utilizará la instrucción ON/GOTO para bifurcar hacia las líneas 200, 300 ó 400 y presentar los pronósticos 1, X ó 2, respectivamente.
La escritura de esta rutina omitiendo la instrucción ON/GOTO no plantea excesivos problemas. El método más inmediato es recurrir al empleo de sucesivas decisiones simples del tipo IF/THEN. Desde luego, el programa será más extenso, pero mantendrá su eficacia. Aquí está una posible solución que consiste, sencillamente, en sustituir la línea 110 del programa inicial por las tres siguientes:

```
110 IF X=1 THEN GOTO 200
120 IF X=2 THEN GOTO 300
130 IF X=3 THEN GOTO 400
```

Al observar el funcionamiento de la primera instrucción del programa, salta a la vista que el valor de X siempre va a ser 1, 2 ó 3. De ahí que sea suficiente con detectar dos de los posibles valores, por ejemplo 2 y 3. Naturalmente, de no adoptar el valor 2 ó 3, X será igual a 1; en consecuencia, puede eliminarse la comparación con 1, de tal forma que su ejecución ocurra en el caso de no resultar positivas las otras dos detecciones.
Una vez eliminada la comparación de X con 1 —asumiendo que X no va a exceder en ningún caso del margen de valores esperado—, el programa quedará como sigue:

```
100 X=(RND*3)+1
110 IF X=2 THEN GOTO 300
120 IF X=3 THEN GOTO 400
200 PRINT "PRONOSTICO: 1"
210 GOTO 600
300 PRINT "PRONOSTICO: X"
310 GOTO 600
400 PRINT "PRONOSTICO: 2"
600 PRINT "¡SUERTE Y A POR LOS CATORCE!"
700 END
```

Su desarrollo es ilustrado por el correspondiente diagrama de flujo. Algunos ordenadores admiten una segunda alternativa. Para ello, han de cumplirse dos condiciones. En primer lugar, su traductor BASIC ha de permitir el empleo de variables y expresiones en el argumento de la instrucción GOTO; por otra parte, debe ser posible encontrar una relación matemática entre las diferentes líneas a las que se desea bifurcar, en función de los valores que tome la variable o expresión a evaluar.
En el ejemplo propuesto, la línea 110 —ocupada por la instrucción ON/GOTO— podría reemplazarse por la siguiente:

```
110 GOTO (X*100)+100
```

Tal como se observa, los posibles valores de X (1, 2 ó 3), provocarán el salto a las

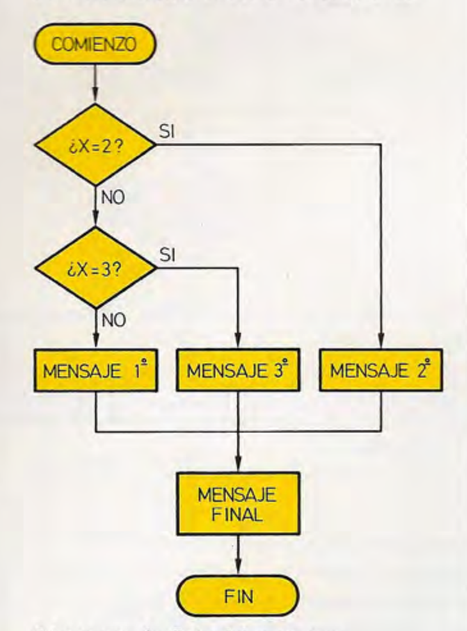
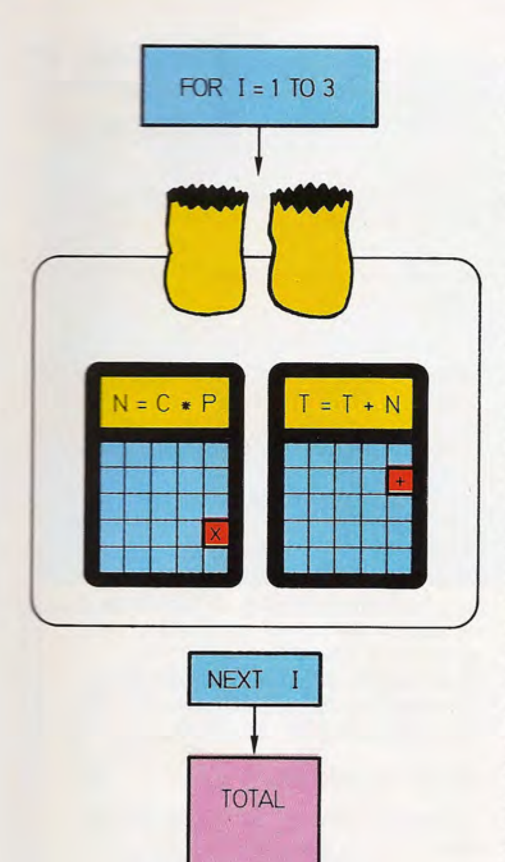


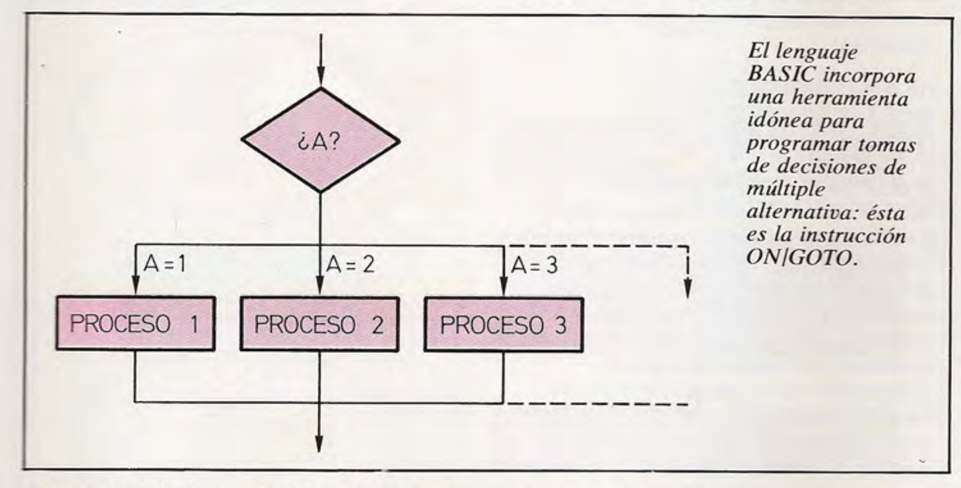
Diagrama de flujo del programa propuesto, una vez sustituida la instrucción ON/GOTO por dos tomas de decisión del tipo IF/THEN.

líneas 200, 300 y 400, respectivamente. Esta es una solución bastante más cómoda; sin embargo, hay que tener en cuenta que no siempre es posible aplicarla.



Aplicando la instrucción FOR/NEXT al ejemplo inicial, se reducirá la amplitud del programa adecuado para el cálculo del importe total de una compra. Un mismo bucle de programa se ocupará de evaluar el importe de cada artículo de acuerdo a la cantidad adquirida en cada caso.





El lenguaje BASIC incorpora una herramienta idónea para programar tomas de decisiones de múltiple alternativa: ésta es la instrucción ON/GOTO.

valor actual de I, para verificar si supera o no al valor final establecido (TO 3).
Tras esta operación caben dos alternativas. Si el nuevo valor de la variable I es inferior al valor final precisado, se repite de nuevo la ejecución del bucle. Por el contrario, si la variable contadora supera el valor final, no se repetirá la ejecución del bucle, sino que el ordenador pasará a ejecutar la instrucción que sigue al NEXT que pone fin a la estructura cíclica.
Cabe recordar que la zona STEP puede omitirse (cosa que ocurre en el ejemplo propuesto). En tal caso, el ordenador asume que el valor del incremento es la unidad.
El valor del incremento puede ser cualquiera, incluso negativo, con lo que se convertiría en un decremento del valor de la variable contadora. En este último caso, la cifra colocada como valor inicial ha de ser mayor que la correspondiente al valor final.
A continuación aparecen dos ejemplos que definen distintos incrementos para la variable contadora.
El primer programa establece un bucle controlado por la variable X. El bucle debe repetirse sucesivamente hasta que X adopte un valor superior a 1.5, teniendo en cuenta que tras completar cada ciclo, el valor de X se incrementará en 0.1 unidades. (Tal como es habitual en BASIC, la coma decimal se sustituye por el punto de acuerdo a la nomenclatura inglesa.)

La ejecución revela que el bucle se recorre en seis ocasiones, mostrando en cada caso el correspondiente valor de la variable X.

```
10 FOR X=1 TO 1.5 STEP 0.1
20 PRINT X;
30 NEXT X
40 END
RUN
1 1.1 1.2 1.3 1.4 1.5
■
```

En el segundo ejemplo, la variable contadora I toma el valor inicial 5 y en cada ciclo sucesivo irá decrementándose en una unidad (STEP −1), hasta que su valor sea inferior a 2 (TO 2). El bucle incluye, sencillamente, una instrucción PRINT que presenta en la pantalla los sucesivos valores de I multiplicados por 4.

```
10 FOR I=5 TO 2 STEP -1
20 PRINT I*4
30 NEXT I
40 END
RUN
20
16
12
4
■
```

No terminan aquí las posibilidades de la estructura FOR/NEXT. A lo largo de la obra se irá observando su versatilidad en la programación de estructuras cíclicas de muy diverso tipo.
Una de las opciones que otorga aún mayor flexibilidad a esta estructura, deriva de

ON/GOTO

Dependiendo del resultado de la expresión (1, 2, 3...) se producirá un salto a la línea cuyo número aparezca en la posición correspondiente (primera, segunda, tercera...) dentro de la lista de números de línea que sigue a GOTO.

Formato: ON <expresión> GOTO <lista de números de línea>

Ejemplos: ON A GOTO 100, 200, 230, 400, 160

la posibilidad de sustituir alguno de los parámetros (valor inicial, valor final o incremento), o todos ellos, por una variable; el programa siguiente constituye un ejemplo elemental aunque ilustrativo de tal posibilidad:

```
 5 LET A=2
10 INPUT B
20 FOR I=A TO B
30 PRINT I
40 NEXT I
50 END
```

Tanto el valor inicial como el final lo aportan variables (A y B, respectivamente), cuyo valor puede asignarse en cualquier zona previa del programa.

LA INSTRUCCION ON/GOTO

En ciertos casos, es necesario plantear decisiones con más de una alternativa. En la vida real, ello no plantea dificultades; sin embargo, en un ordenador la cosa no es tan fácil. Suponga, por ejemplo, que disponemos de un menú con varias opciones:

1. Imprimir las máximas puntuaciones.
2. Realizar una demostración.
3. Empezar el juego.

Un menú de tres opciones que conduce a tres procesos distintos. Según el número de la opción elegida, el ordenador habrá de ejecutar una serie de operaciones, muy distintas en cada caso. Uno de los posibles métodos para programar esta toma de decisión en BASIC consiste, simplemente, en encadenar sucesivas instrucciones IF/THEN; por ejemplo:



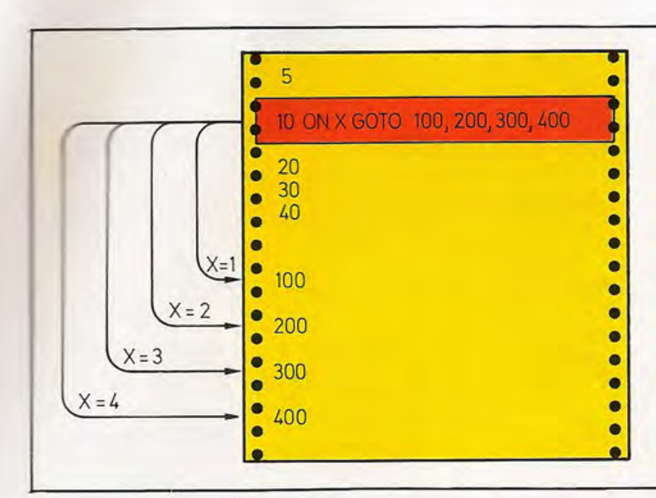
TABLA DE CONVERSION

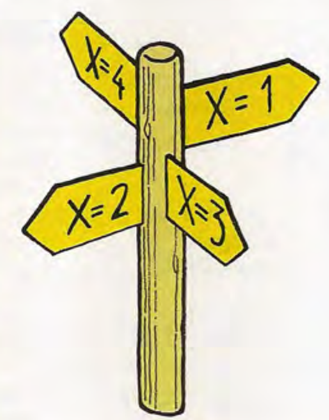
ORDENADOR	FOR/NEXT		ON/GOTO
	FOR <variable>=<i> TO <f> STEP <r>	**NEXT <variable>**	**ON <expresión> GOTO <lista>**
APPLE II (APPLESOFT)	FOR <variable>=<i> TO <f> STEP <r>	NEXT <variable>	ON <expresión> GOTO <lista>
APRICOT (M-BASIC)	FOR <variable>=<i> TO <f> STEP <r>	NEXT <variable>	ON <expresión> GOTO <lista>
ATARI	FOR <variable>=<i> TO <f> STEP <r>	NEXT <variable>	ON <expresión> GOTO <lista>
CBM 64	FOR <variable>=<i> TO <f> STEP <r>	NEXT <variable>	ON <expresión> GOTO <lista>
DRAGON	FOR <variable>=<i> TO <f> STEP <r>	NEXT <variable>	ON <expresión> GOTO <lista>
EQUIPOS MSX	FOR <variable>=<i> TO <f> STEP <r>	NEXT <variable>	ON <expresión> GOTO <lista>
HP-150	FOR <variable>=<i> TO <f> STEP <r>	NEXT <variable>	ON <expresión> GOTO <lista>
IBM PC	FOR <variable>=<i> TO <f> STEP <r>	NEXT <variable>	ON <expresión> GOTO <lista>
MPF	FOR <variable>=<i> TO <f> STEP <r>	NEXT <variable>	ON <expresión> GOTO <lista>
NCR DM-V (MS-BASIC)	FOR <variable>=<i> TO <f> STEP <r>	NEXT <variable>	ON <expresión> GOTO <lista>
NEW BRAIN	FOR <variable>=<i> TO <f> STEP <r>	NEXT <variable>	ON <expresión> GOTO <lista>
ORIC	FOR <variable>=<i> TO <f> STEP <r>	NEXT <variable>	ON <expresión> GOTO <lista>
SHARP MZ-700 (MZ-BASIC)	FOR <variable>=<i> TO <f> STEP <r>	NEXT <variable>	ON <expresión> GOTO <lista>
SINCLAIR QL	FOR <variable>=<i> TO <f> STEP <r>	NEXT <variable>	ON <expresión> GOTO <lista>
SPECTRAVIDEO	FOR <variable>=<i> TO <f> STEP <r>	NEXT <variable>	ON <expresión> GOTO <lista>
ZX-SPECTRUM	FOR <variable>=<i> TO <f> STEP <r>	NEXT <variable>	ON <expresión> GOTO <lista>

<variable>: variable numérica. <i>: expresión que aporta el valor inicial. <f>: expresión que define el valor final. <r>: expresión que define el incremento. <lista>: lista de números de línea.

FORMULACIONES DE LOS COMANDOS

FOR <variable>: <i> TO <f> STEP <r>: define un bucle que se ejecutará repetidamente hasta que la variable alcance el valor de f, en sucesivos incrementos de valor r. NEXT <variable>: cierra el bucle definido por medio de una instrucción FOR. ON <expresión> GOTO <lista>: transfiere la secuencia de ejecución a uno de los números de línea que figuran en la lista, en función del valor de la expresión.





Funcionamiento de la instrucción ON/GOTO. El valor que adopte la variable X o, en general, la expresión que sigue a ON, determinará el número de línea al que debe bifurcar el programa. Los números de línea que pueden verse afectados son los que aparecen en la lista incluida en la zona GOTO.

```
10 INPUT A
20 IF A=1 THEN GOTO 500
30 IF A=2 THEN GOTO 1000
40 IF A=3 THEN GOTO 1500
50 PRINT "ESA OPCION NO ESTA EN EL
   MENU"
60 GOTO 10
■
```

El bloque de instrucciones empieza captando el número de la opción elegida a través de la instrucción INPUT A (línea 10). Acto seguido, las instrucciones 20, 30 y 40 detectan cada una de las tres posibles alternativas, y derivan la secuencia de ejecución hacia los subprogramas correspondientes a cada una de ellas. En efecto, se ha supuesto que los respectivos subprogramas se encuentran situados a partir de las líneas 500, 1000 y 1500.

El ejemplo termina con las instrucciones 50 y 60. La primera de ellas presentará un mensaje al usuario en el caso de que se introduzca un número que no corresponda a una de las opciones permitidas; tras ello, y por efecto de la instrucción GOTO 10, el ordenador aguardará a que se introduzca de nuevo el número de la opción.

De la eficacia de esta solución no cabe la menor duda. No obstante, ¿qué sucederá si el número de opciones del menú se eleva a veinte o treinta? El programa puede prolongarse más y más, a base de ir colocando sucesivas instrucciones de tipo IF/THEN, y todo ello con el único objetivo de detectar la opción elegida y bifurcar la ejecución a la línea adecuada.

En efecto, el método es eficaz, pero dudosamente práctico cuando las alternativas son múltiples. Hay que buscar otro procedimiento más cómodo y rápido.

Para programar este tipo de tomas de decisión de una forma más razonable, es necesario recurrir a estructuras lógicas más complejas. Estructuras que permitan seleccionar una entre varias opciones, dependiendo del valor que adopte una expresión a evaluar.

El lenguaje BASIC posee una instrucción especializada en tal cometido; ésta es ON/GOTO, cuyo formato general es el siguiente:

ON <expresión> GOTO <lista de números de línea>

Al utilizar la nueva instrucción, el programa anterior adoptará un nuevo aspecto:

```
10 INPUT A
20 ON A GOTO 500, 1000, 1500
30 PRINT "ESA OPCION NO ESTA EN EL
   MENU"
40 GOTO 10
■
```

Sin lugar a dudas, el empleo de la instrucción ON/GOTO reduce la amplitud del programa; especialmente si pensamos en un caso en el que la decisión esté sujeta a un elevado número de opciones.

Las dos zonas que conforman a la nueva instrucción están perfectamente delimitadas. La primera de ellas la ocupa el comando ON y su argumento. Este contiene la expresión a evaluar, o sencillamente la variable cuyo valor condicionará la bifurcación o salto.

El comando GOTO y su argumento, integrado por una serie de números de línea separados por comas, dan cuerpo a la segunda zona de la instrucción. La relación entre ambas es muy simple. Tras evaluar el resultado de la expresión que acompaña a ON, el ordenador ejecutará un salto al número de línea que sigue a GOTO, cuya posición coincida con el valor extraído en la zona ON. Esto es: si el valor de la expresión es 1, el salto se producirá al primer número de línea; si es 2 al segundo; si su valor es 3 al tercero y así sucesivamente.

En el programa ejemplo, cuando se elija la opción 1 —tal será el valor de la variable A que ocupa la zona ON—, el salto conducirá a la línea 500; si se elije la opción 2, el salto llevará a la línea 1000; mientras que si la opción seleccionada es la 3, el programa bifurcará la línea 1500.

En definitiva, la elección se realiza a partir del valor de la expresión localizada en la zona ON. Si fuera necesario, el ordenador redondeará el valor del resultado al número entero más próximo, para que de esta forma quede precisado el orden del número de línea al que debe producirse el salto.

También puede ocurrir que el valor de la expresión sea cero, o mayor que el número de elementos en la lista; en tales circunstancias no se producirá salto alguno, sino que se ejecutará la instrucción siguiente a ON/GOTO.

En otro orden, puede suceder que el valor de la expresión sea negativo superior a 255; ambas situaciones harán que el ordenador presente un mensaje de error en la pantalla.

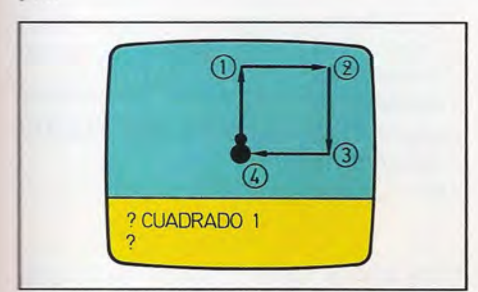
Logo (7)

TURTLE GRAPHICS: procedimientos con la tortuga

Dentro del apartado precedente dedicado al "Turtle graphics" (Logo 3), se presentaron los fundamentos del tratamiento de gráficos con el LOGO. Asimismo, en otro punto de la obra se detalló el concepto y uso de los procedimientos. Este capítulo va a combinar ambos temas, trasladando las posibilidades que ofrecen los procedimientos al terreno de la confección de dibujos con la tortuga. Se crearán procedimientos capaces de trazar gráficos y se verán las enormes posibilidades que éstos aportan a la confección de graficos.

PROCEDIMIENTOS CON LA TORTUGA

Uno de los ejemplos incluidos en la presentación del "Turtle graphics" fue la secuencia de órdenes adecuadas para trazar una línea en la zona superior de la pantalla, devolviendo a la tortuga, a continuación, a su posición original. El referido conjunto de órdenes puede integrarse dentro de un procedimiento, sin más que apelar al método de definición ya estudiado; por ejemplo:

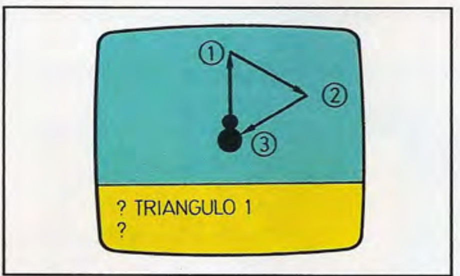


Dibujo creado en la pantalla por medio del procedimiento CUADRADO 1. Los vértices numerados corresponden a los puntos que alcanza la tortuga tras ejecutar cada una de las sucesivas órdenes FORWARD 50.

```
TO LINEA
CS
PENUP
FORWARD 50
RIGHT 90
PENDOWN
FORWARD 50
PENUP
HOME
END
```

Al integrar la secuencia de órdenes dentro de un procedimiento, se evita la necesidad de teclear de nuevo las mismas órdenes cada vez que desee realizarse la misma acción.
Todas las posibilidades de la creación de gráficos por medio de la tortuga pueden incluirse en procedimientos. A continuación, se describen algunos procedimientos con la tortuga que pueden resultar útiles a la hora de trazar dibujos más complejos.
El primero de ellos, cuyo nombre es CUADRADO 1, es capaz de dibujar un cuadrado de 50 posiciones de lado. Tras confeccionar el dibujo, la tortuga queda emplazada en su posición de origen, tal como muestra la correspondiente pantalla.

```
TO CUADRADO1
FORWARD 50
RIGHT 90
FORWARD 50
RIGHT 90
FORWARD 50
RIGHT 90
FORWARD 50
RIGHT 90
END
```



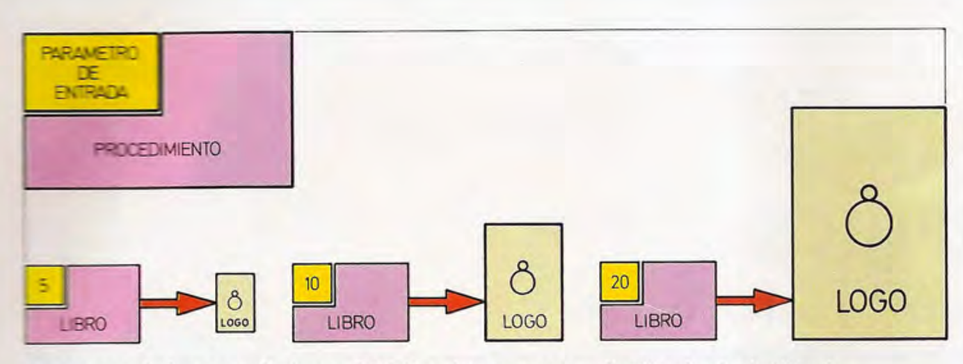
El procedimiento TRIANGULO 1 instruye a la tortuga para que trace en la pantalla el dibujo ilustrado. La ejecución de cada una de las tres órdenes FORWARD que en él intervienen conduce a la tortuga a los distintos vértices del triángulo.

Un nuevo ejemplo lo aporta el procedimiento que sigue (TRIANGULO1); su ejecución dibuja un triángulo cuyo lado es de 50 posiciones.

```
TO TRIANGULO1
FORWARD 50
RIGHT 120
FORWARD 50
RIGHT 120
FORWARD 50
RIGHT 120
END
```

De nuevo, en este segundo ejemplo, una vez realizado el dibujo, la tortuga queda en la posición de partida. Cabe notar que la suma de los ángulos girados es de 360 grados (una vuelta completa), de ahí que el retorno de la tortuga a la posición de origen, y con la orientación adecuada, sea directo.
Ahora, es posible utilizar los dos procedimientos elementales definidos para crear figuras más complicadas. Un ejemplo lo constituye el procedimiento BANDERIN1 que hace uso de TRIANGULO1 para dibujar un banderín con mástil.

```
TO BANDERIN1
FORWARD 30
TRIANGULO1
END
```



La técnica de los procedimientos LOGO es íntegramente aplicable al método "Turtle Graphics". Las secuencias de órdenes de dibujo pueden adoptar la forma de procedimientos, con o sin parámetros.

El cuadro adjunto contiene tres nuevos ejemplos de procedimientos de dibujo más evolucionados (CASA, ROMBO y MOLINO), que hacen uso de los ya definidos: CUADRADO1 y TRIANGULO1.
Estas son algunas posibilidades que brinda la aplicación de los procedimientos LOGO al método "Turtle graphics". Por supuesto, la versatilidad de los procedimientos con parámetros también es trasladable al caso de la tortuga. Muestra de ello van a constituirla los ejemplos de los próximos apartados.

PROCEDIMIENTOS A MEDIDA

Una vez creados los procedimientos CUADRADO1 y TRIANGULO1 quedan definidos ambos dibujos en el campo de trabajo. Para utilizarlos, no hay más que llamar al respectivo procedimiento utilizando su nombre. Las referidas figuras pueden trasladarse a cualquier posición de la pantalla. Para ello basta con colocar la tortuga en la posición y orientación adecuadas. E incluso puede alterarse el tamaño de los dibujos. Esto último exige algunos retoques en el procedimiento para que sea posible introducir la longitud de un lado. Tal valor puede introducirse como parámetro de entrada al procedimiento. Veamos cuál es el aspecto de ambos ejemplos una vez convertidos en procedimientos con un parámetro de entrada.

```
TO CUADRADO :L
FORWARD :L
RIGHT 90
FORWARD :L
RIGHT 90
FORWARD :L
RIGHT 90
FORWARD :L
RIGHT 90
END
```

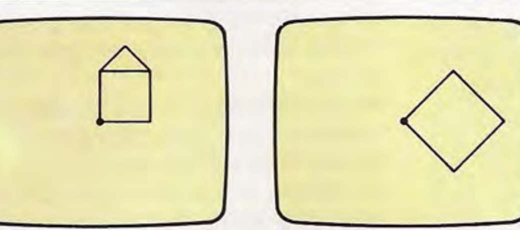
En el primer caso (dibujo de un cuadrado), la diferencia aparece en el parámetro L. Su cometido es definir la magnitud del lado del cuadrado, de ahí que su valor acompañe a las cuatro órdenes FORWARD.
Análogamente, puede definirse un procedimiento capaz de dibujar un triángulo cuyo lado lo precisará el parámetro de entrada L.

```
TO TRIANGULO :L
FORWARD :L
RIGHT 120
FORWARD :L
RIGHT 120
FORWARD :L
RIGHT 120
END
```

En definitiva, los dos nuevos procedimientos permiten definir el tamaño de la figura a través del valor o parámetro que se agrega como entrada al procedimiento. El siguiente ejemplo, CUADROS, utiliza esta posibilidad para dibujar en la pantalla un conjunto de cuadrados de distinto lado. CUADROS no es más que un procedimiento evolucionado, creado a partir de la repetición del procedimiento con parámetro CUADRADO :L.

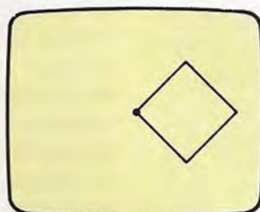
```
TO CUADROS
CUADRADO 20
CUADRADO 40
CUADRADO 60
CUADRADO 80
CUADRADO 100
END
```

De la misma forma se pueden definir procedimientos más genéricos. Por ejemplo, uno que dibuje polígonos regulares dando el número de lados y su longitud; tal es el caso del procedimiento POLIGONO definido a continuación:



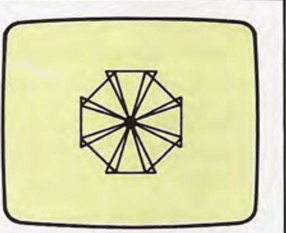
CASA

```
TO CASA
CUADRADO1
FORWARD 50
RIGHT 30
TRIANGULO1
END
```



ROMBO

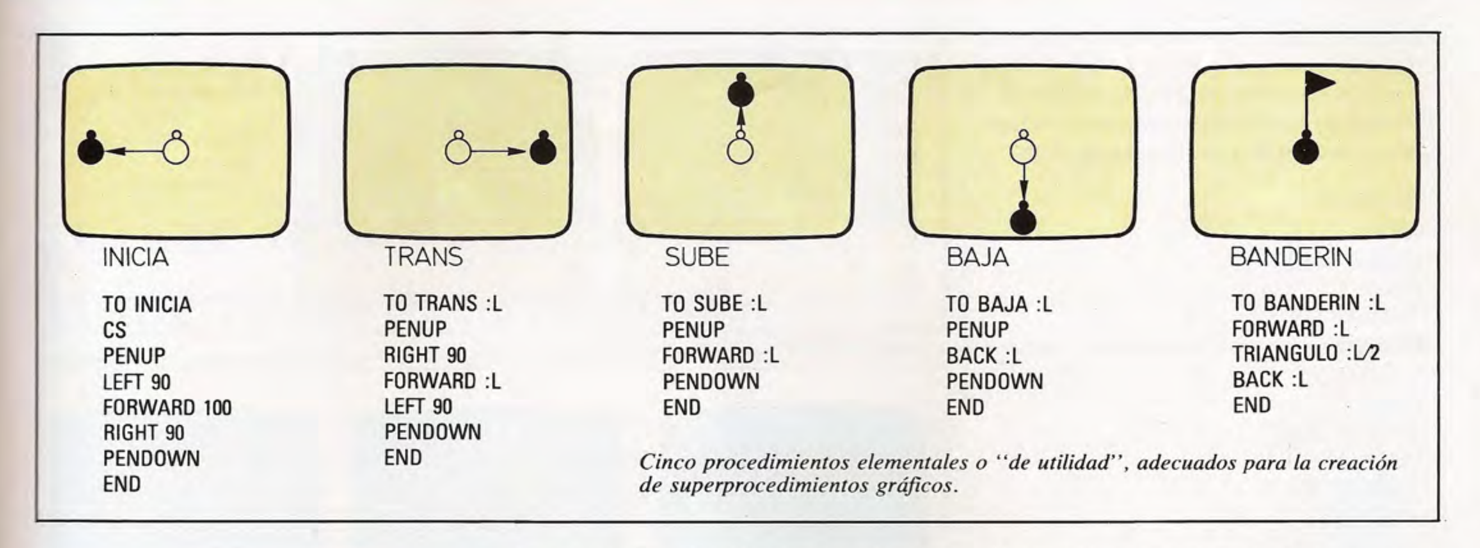
```
TO ROMBO
RIGHT 30
TRIANGULO1
RIGHT 60
PX
TRIANGULO1
PENDOWN
END
```



MOLINO

```
TO MOLINO
TRIANGULO1
LEFT 45
TRIANGULO1
LEFT 45
TRIANGULO1
LEFT 45
TRIANGULO1
LEFT 45
TRIANGULO1
LEFT 45
TRIANGULO1
LEFT 45
TRIANGULO1
LEFT 45
TRIANGULO1
LEFT 45
END
```

La aplicación de la técnica de los procedimientos LOGO al dibujo con la tortuga, permite construir procedimientos gráficos, cada vez más complejos, al asociar otros procedimientos elementales. Las pantallas ilustran la ejecución de tres procedimientos en cuya definición intervienen otros más primitivos (CUADRADO 1 y TRIANGULO 1) definidos con anterioridad.



Cinco procedimientos elementales o "de utilidad", adecuados para la creación de superprocedimientos gráficos.

```
TO POLIGONO :N :L
REPEAT :N [FORWARD :L RIGHT 360/:N]
END
```

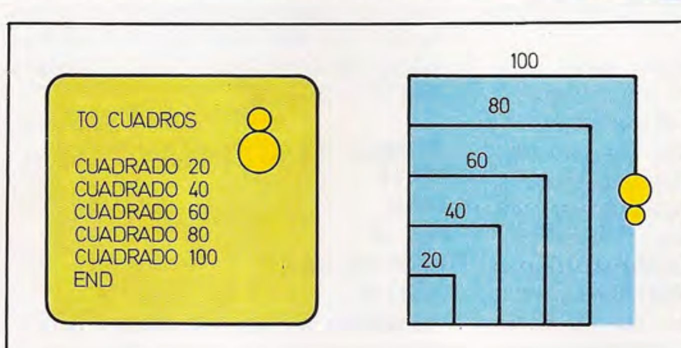
Su definición, extremadamente compacta, adopta el aspecto de procedimiento con dos parámetros de entrada: N (número de lados del polígono) y L (longitud del lado). Un aspecto a señalar es la presencia dentro del procedimiento del comando REPEAT. Su estudio detallado tendrá lugar en el capítulo dedicado a "bucles"; por el momento basta con saber que ordenará la repetición de las acciones encerradas entre corchetes tantas veces como indique el parámetro N.
Si el número de lados especificado es 3 ó 4, POLIGONO se comporta como los procedimientos TRIANGULO o CUADRADO, respectivamente.

COLECCION DE EJEMPLOS

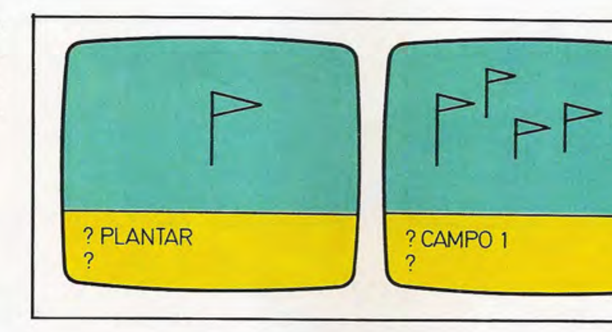
Para trasladar al terreno práctico los conocimientos adquiridos hasta el momento, vamos a poner a trabajar a la disciplinada tortuga.

La siguiente colección de ejemplos hará uso de los procedimientos elementales o de utilidad INICIA, TRANS, SUBE, BAJA y BANDERIN que se detallan en el cuadro correspondiente. Los referidos procedimientos de utilidad entrarán a formar parte de la definición de superprocedimientos más evolucionados, capaces de "instruir" a la tortuga para que trace en la pantalla dibujos más elaborados.
Al utilizar los cinco procedimientos elementales, definidos en el cuadro adjunto, nuestro simpático dibujante —la tortuga— debe estar orientada en sentido vertical y "mirando" hacia el borde superior de la pantalla.
Con esta precaución, la actuación de los referidos procedimientos elementales será la que sigue:
INICIA: desplaza a la tortuga a la izquierda de la pantalla, levantando la tiza para que no deje trazo al trasladar su origen.
TRANS: traslada a la tortuga tantas posiciones a la derecha como indique el valor del parámetro L; de nuevo, sin dejar trazo alguno en la pantalla.
SUBE, BAJA: ambos procedimientos permiten alterar la posición vertical de la tortuga: hacia arriba o hacia abajo, respectivamente.
BANDERIN: dibuja un banderín del tamaño especificado.
Con estos procedimientos elementales, van a dibujarse a continuación varias escenas, sencillas aunque plenamente ilustrativas de las posibilidades que otorgan los procedimientos al trazado de gráficos en pantalla. Desde luego, el usuario puede crear sus propios dibujos utilizando y alterando alguno de los siguientes ejemplos.
El primero de los ejemplos (PLANTAR), hace uso de los procedimientos SUBE, BANDERIN y BAJA para dibujar un banderín en la zona superior de la pantalla.

```
TO PLANTAR :L
SUBE :L
BANDERIN 100-(:L/3)
BAJA :L
END
```



Ejecución del superprocedimiento CUADROS. Dentro de su definición interviene repetidamente un procedimiento (CUADRADO) cuyo parámetro de entrada coincide con el lado del cuadrado a dibujar.



Resultado de la ejecución de dos procedimientos de dibujo definidos en el texto: PLANTAR y CAMPO 1.







A su vez, CAMPO1, es un procedimiento evolucionado que recurre a los procedimientos elementales INICIA, BANDERIN, TRANS y PLANTAR para dibujar en la pantalla una escena con cinco banderines:

```
TO CAMPO1
INICIA
BANDERIN 60
TRANS 50
PLANTAR 60
TRANS 40
PLANTAR 20
TRANS 60
PLANTAR 35
TRANS 40
PLANTAR 53
END
```

A continuación, se desarrollan nuevos ejemplos a partir de POLIGONO. El procedimiento CIRCUNFERENCIA particulariza el procedimiento POLIGONO, definido en el apartado anterior, para un número grande de lados (30) y calcula su longitud partiendo del radio. Este procedimiento se utiliza dentro de ARBOL para dibujar la copa.

```
TO ARBOL :L
FORWARD :L
LEFT 90
CIRCUNFERENCIA :L*3/4
RIGHT 90
BACK :L
END
```

```
TO CIRCUNFERENCIA :R
POLIGONO 30 ((:R*6.28)/30)
END
```

A su vez, BOSQUE se apoya en el procedimiento ARBOL, situando cada elemento en la pantalla con la ayuda del procedimiento PON.

```
TO PON :L
SUBE :L
ARBOL (100-:L)/3
BAJA :L
END
```

Este último es similar a PLANTAR, aunque recurriendo a ARBOL en lugar de a BANDERIN. Su empleo repetido dentro del procedimiento BOSQUE da lugar a la representación gráfica que muestra la correspondiente pantalla.

```
TO BOSQUE
INICIA
PON 50
TRANS 40
PON 20
TRANS 40
PON 35
TRANS 40
PON 17
TRANS 40
PON 65
TRANS 40
PON 44
END
```

Cada vez los dibujos pueden ser más y más complejos, al irse enriqueciendo con la incorporación de procedimientos definidos a base de otros más elementales.
Por ejemplo, introduciendo algunas variantes al procedimiento INICIA puede definirse un nuevo procedimiento que denominaremos INI; esta vez, el desplazamiento de la tortuga no borrará el contenido de la pantalla.

```
TO INI
PENUP
HOME
LEFT 90
FORWARD 120
RIGHT 90
PENDOWN
END
```

Asociando los procedimientos BOSQUE, BAJA y CASA es posible ya el dibujo de un paisaje en el que aparezcan árboles e incluso una casa.

```
TO PAISAJE
BOSQUE
BAJA 30
CASA
PENUP
HOME
PENDOWN
END
```

La evolución puede seguir en sucesivas etapas. Por ejemplo, introduciendo el procedimiento anterior (PAISAJE) dentro de un nuevo superprocedimiento capaz de plasmar en la pantalla el dibujo de un pueblo de la mano de la tortuga LOGO.

```
TO PUEBLO
PAISAJE
BAJA 40
CASA
INI
BAJA 50
CASA
PENUP
HOME
PENDOWN
END
```

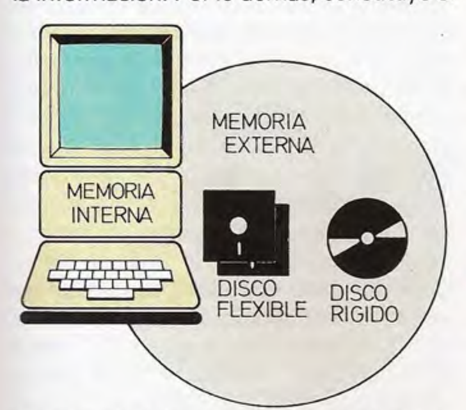
Estructura de la memoria

¿Cómo reside el CP/M en la memoria del ordenador?

La memoria de un ordenador es el soporte en el que se almacenan los programas y datos: o lo que es lo mismo, el lugar donde residen todos los elementos *software* necesarios para el funcionamiento de la máquina: el sistema operativo, los programas de aplicación y los datos necesarios para la ejecución de los programas. Esta es una definición genérica y aplicable a cualquier ordenador.
En la práctica, una de las diferencias que se manifiestan entre los distintos ordenadores reside en la forma en la que cada uno de ellos distribuye los elementos software en su memoria. Ello se debe, básicamente, a los condicionantes que impone el propio sistema operativo que rige la actividad del ordenador.
La memoria de un ordenador puede dividirse en dos grandes apartados.

* *Memoria primaria*
También denominada memoria central o residente. Está alojada en el interior de la máquina y consta de un conjunto de circuitos electrónicos capaces de almacenar la información. Por lo demás, constituye el soporte básico de la información almacenada en el ordenador.

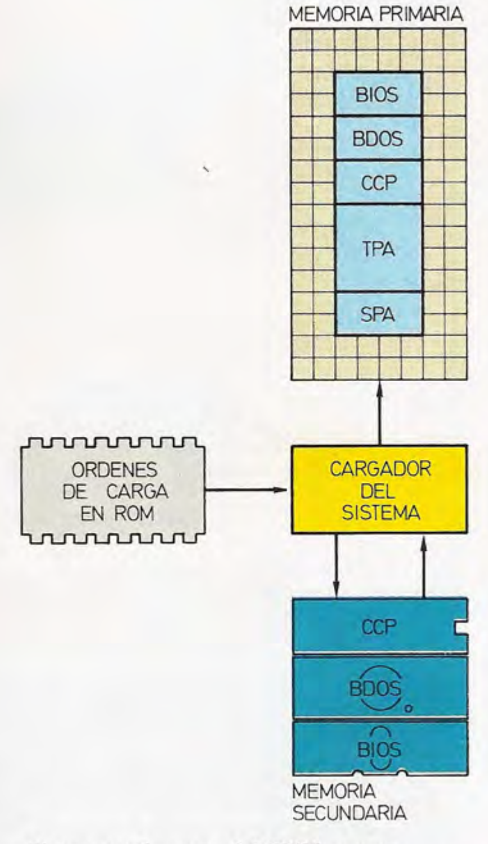
* *Memoria secundaria*
Está localizada fuera del ordenador y los dispositivos en los que se apoya entran dentro de la categoría de los periféricos. El soporte de almacenamiento suele ser un disco magnético, ya sea de tipo flexible o rígido. Su presencia, en número y capacidad de almacenamiento, depende del modelo específico de ordenador. Algunos ordenadores personales modernos, como el IBM-PC, el DECISION MATE V de NCR o el PC de Ericcson, suelen complementarse con un disco rígido de 10 Mbytes y una unidad para discos flexibles (floppy disk) de aproximadamente 350 Kbytes por disco.
Tal y como propagan estas cifras, una diferencia fundamental entre los soportes físicos que pueden formar parte de la memoria secundaria o memoria de masa del ordenador, es la densidad de información que pueden almacenar.
El sistema operativo CP/M puede encontrarse almacenado, dependiendo del estado de operación del ordenador, en la memoria primaria o bien en la secundaria. Por supuesto, su lugar habitual es la memoria secundaria; no obstante, cuando el ordenador debe utilizar alguna de sus funciones, es imprescindible que antes lo traslade a la memoria central. No hay que perder de vista que la máquina sólo puede procesar la información que en cada instante reside en su memoria interna.



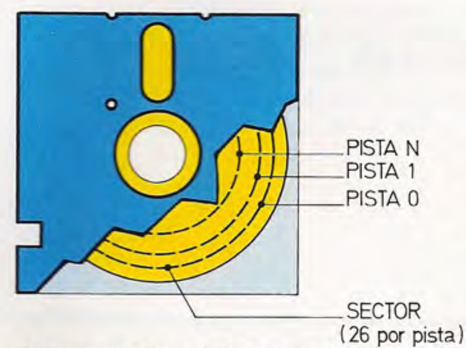
La memoria del ordenador cabe dividirla en dos grandes apartados: memoria central, interna o residente en la unidad central y memoria externa o de masa. Esta última la aportan al sistema los periféricos de almacenamiento conectados al mismo.

ESTRUCTURA DEL ALMACENAMIENTO EN DISCO

En el instante de conectar al ordenador, entran en juego un grupo de instrucciones que residen permanentemente en su memoria central; instrucciones cuya ejecución hará que la máquina inicie automáticamente el proceso de lectura de la memoria de masa (normalmente disco) en la que se encuentra almacenado el sistema operativo. En ciertos modelos, es posible que la carga no se realice automáticamente, sino que, en su lugar, la máquina comunique al usuario que está dispuesta a realizar dicha lectura. Ante esta situa-

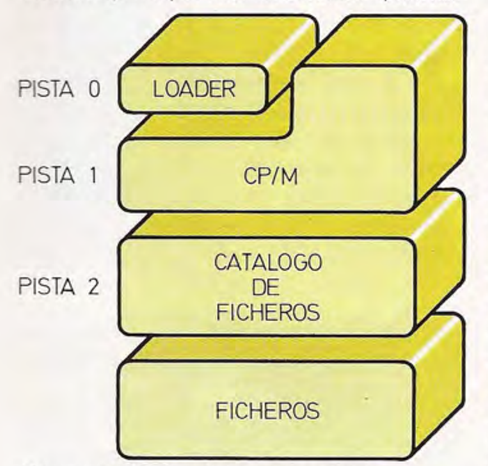


El proceso de carga del CP/M supone trasladar a la memoria central del ordenador los módulos del sistema operativo residentes en la memoria secundaria. La operación es iniciada por las órdenes de carga procedentes de unidad central.



Disposición física de las pistas de grabación en la superficie de un disco flexible.

ción, el usuario debe introducir el disco con el sistema operativo en la unidad de lectura principal del ordenador (normalmente designada con las letras "A" o "S") y dar la orden adecuada para que empiece la carga.



Estructura básica de las pistas del disco flexible que almacena al sistema operativo CP/M.

Si se estuvieran utilizando discos flexibles de simple densidad, con 26 sectores por pista, la localización habitual de los diferentes bloques del sistema operativo sobre el disco coincidirá con la que se detalla en los próximos párrafos.
La primera operación de lectura afecta al *cargador* o "loader", situado en el sector 1 de la pista 0. Su misión es orquestar la carga de los tres bloques del sistema operativo y asignarles sus respectivas posiciones en la memoria primaria del ordenador. El cargador es, en definitiva, un programa que empieza a ejecutarse en el momento en el que pasa a residir en la memoria primaria, y que toma el control del proceso de carga de los siguientes elementos software.
Tras el cargador, y ocupando lo que resta de la pista 0 y toda la pista 1 del disco, se encuentran los tres módulos del sistema operativo: CCP, BDOS y BIOS, en ese mismo orden. La respectiva distribución de las pistas es la que refleja el gráfico adjunto. Las restantes pistas del disco se destinan a ficheros propios del usuario; si bien, se reserva la pista 2 (la primera libre a continuación de las que ocupa el sistema operativo) para el catálogo de ficheros almacenados en el disco.
En el caso del CP/M, el catálogo ocupa un máximo de 32 sectores; esto es, algo más de una pista. A su vez, los sectores están distribuidos en zonas de 32 octetos, de las cuales existen cuatro por cada sector del disco. Las pistas sobre las que se encuentra grabado el sistema operativo —pistas 0 y 1— son pistas exclusivamente reservadas al mismo. En ellas, el sistema operativo CP/M no aparece grabado a modo de ficheros convencionales, sino de una forma particular y única, con la que no será posible que el usuario lo borre inadvertidamente.
Para comprobar este extremo puede ejecutarse el comando DIR. Este presentará en la pantalla la lista de los ficheros residentes en el disco. Acto seguido puede ordenarse el borrado de todos los ficheros para, a continuación, iniciar un nuevo proceso de carga. El usuario observará que dicho proceso se desarrolla con toda normalidad, a pesar de que una llamada al comando DIR señale la inexistencia de ficheros en el disco.

PISTA	SECTOR	MODULO CP/M
00	01	CARGADOR DEL SISTEMA
00 00	02 17	CCP
00 01	18 19	BDOS
01 01	20 26	BIOS
02	01	CATALOGO Y FICHEROS

Distribución de pistas y sectores de pista en el disco flexible en el que reside el sistema operativo CP/M.

ESTRUCTURA DE LA MEMORIA PRIMARIA

El conjunto de la memoria primaria a disposición del ordenador se encuentra repartida, habitualmente, en dos zonas básicas. La primera se reserva al sistema operativo, para que en ella almacene los datos intermedios de su uso exclusivo y realice las operaciones propias de su actividad como supervisor del trabajo global de la máquina.
La segunda zona coincide con el área de memoria destinada al almacenamiento y ejecución de los programas de aplicación. El tamaño de esta última es variable, puesto que depende del volumen total de la memoria que incorpore el ordenador. Al respecto, cabe señalar que el área destinada al sistema operativo es fija e independiente del espacio total.
La distribución específica de ambas zonas depende de la estructura de la memoria central y del propio sistema operativo. Más concretamente, el sistema operativo CP/M divide la memoria en cinco partes, de las cuales se reserva cuatro para su propio uso, mientras que la quinta, denominada TPA (Transient Program Area) la destina a los programas del usuario. Estas porciones de memoria, reflejadas en la figura adjunta, se encuadran en las dos

Velocidad de ejecución de los ordenadores

Desde que se empezaron a fabricar los primeros ordenadores electrónicos, en la década de los 40, el objetivo básico ha sido conseguir una mayor velocidad de trabajo. Esta es una característica dependiente de un factor íntimo de la máquina: el denominado tiempo de ciclo.

El tiempo de ciclo equivale al intervalo que transcurre entre cada dos pulsaciones del reloj maestro que marca el ritmo de operación del ordenador. En consecuencia, a medida que se reduce el tiempo de ciclo, mayor será la velocidad de trabajo de la máquina.

Parece evidente que para conseguir una mayor celeridad en las operaciones es preciso reducir el tiempo de ciclo. Así es, en efecto. Sin embargo, hay una exigencia fundamental a satisfacer: los componentes electrónicos que conforman la intimidad del ordenador y que operan a modo de interruptores, deben ser capaces de "actuar" (cambiar de estado) en un tiempo inferior al de ciclo; de lo contrario no cabe pensar en un ordenador en funcionamiento.

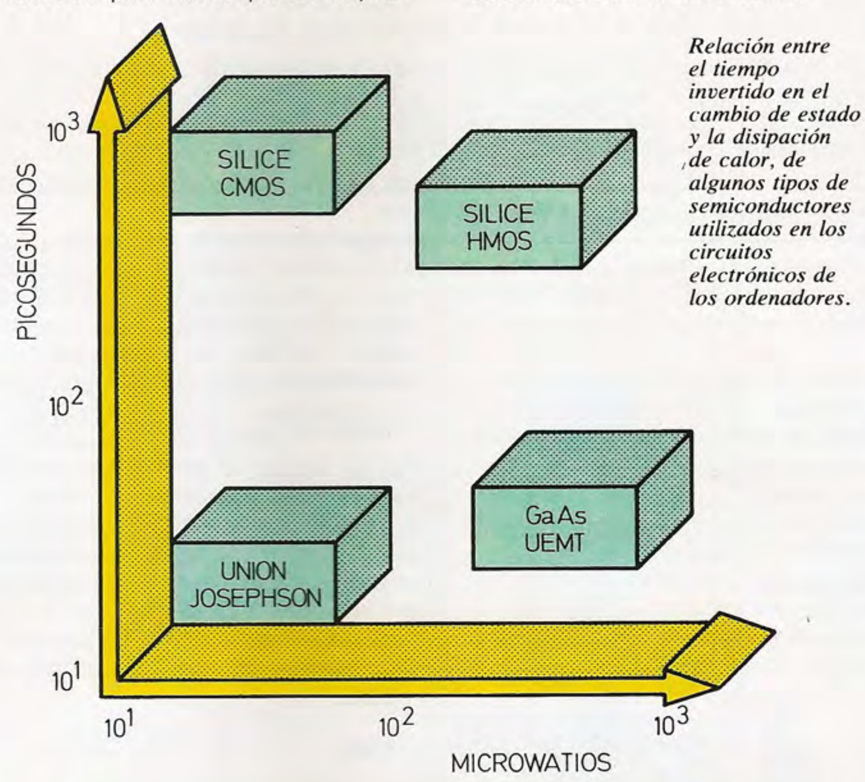
Ahí está el condicionante real de la velocidad del ordenador. En la medida en que sea posible acelerar la velocidad de actuación de semejantes "interruptores", podrá minimizarse el tiempo de ciclo y, en definitiva, acelerar el trabajo del ordenador.

En los albores de la informática moderna, los circuitos del ordenador estaban construidos a base de válvulas de vacío; componentes extremadamente lentos en su función de interruptores. Entre los primeros representantes de esta categoría de máquinas cabe citar a los ordenadores Z.3 y Z.4, puestos en funcionamiento en Berlín en el año 1941 por Konrad Zuse. El tiempo empleado en la operación base del Z.3 era de 0.43 segundos.

El salto espectacular que ha conducido a las veloces máquinas actuales llegó con la tecnología de los semiconductores. Primero fueron los transistores y más tarde los circuitos integrados, los populares "chips". Su presencia revirtió en un drástico aumento de la velocidad de operación. Sin lugar a dudas, el transistor y luego los circuitos integrados, eran "interruptores" bastante más veloces que las válvulas de vacío.

En la actualidad, la aplicación de los últimos hallazgos de la tecnología de los semiconductores, permite crear ordenadores de alto rendimiento, con tiempos de ciclo de 40 a 50 nanosegundos e incluso inferiores. Por ejemplo, el CRAY-1, un ordenador de diseño especial que llega a operar con tiempos de ciclo de 12 nanosegundos.

El salto a tiempos de ejecución aún más reducidos exige nuevas tecnologías. Además de interruptores más rápidos, es preciso reducir el tamaño de los mismos. Este último condicionante choca con el hecho de que a medida que disminuye el volumen, más problemática resulta la disipación del calor generado. Todos ellos son inconvenientes no mayores que los que hace cuarenta años planteaban las válvulas de vacío. No cabe duda, pues, que el avance de la tecnología electrónica saldrá airosa del reto que supone lograr ordenadores cada vez más rápidos.

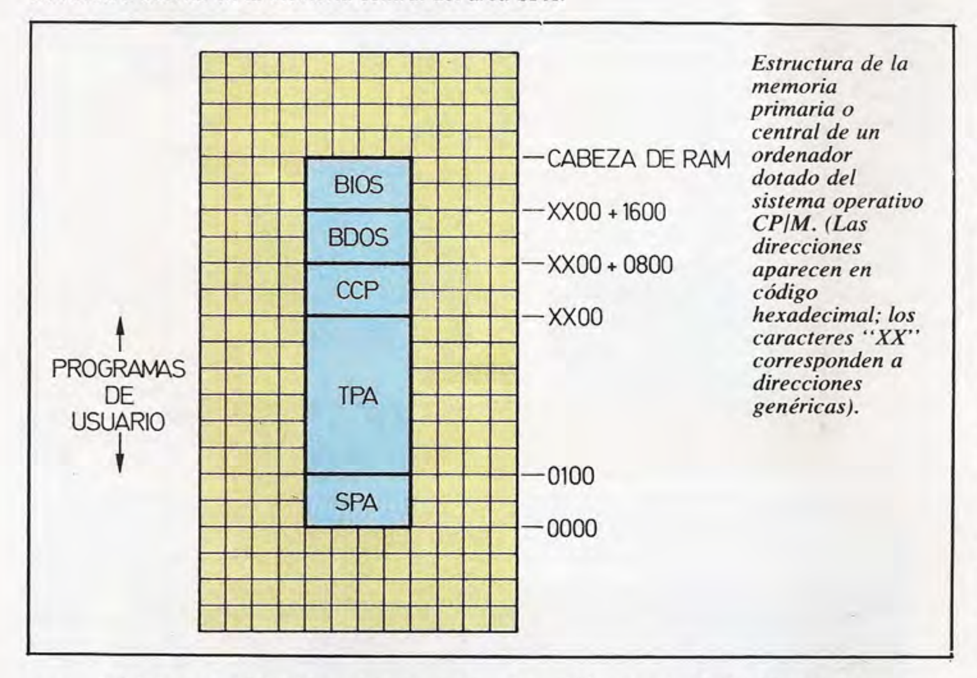


Relación entre el tiempo invertido en el cambio de estado y la disipación de calor, de algunos tipos de semiconductores utilizados en los circuitos electrónicos de los ordenadores.



LOCALIZACION DE MEMORIA	FUNCION
0000 - 0002	BIFURCACION A RUTINA DE ARRANQUE BIOS
0003	10 BYTE
0004	NUMERO DE UNIDAD ACTUAL
0005 - 0007	BIFURCACION AL VECTOR DE ENTRADA BDOS
0008 - 0037	RESERVADO INSTRUCCIONES MAQUINA
0038 - 003F	RST 7, USADO POR DDT
0039 - 003F	RESERVADO INSTRUCCIONES MAQUINA
0040 - 004F	RESERVADO AL MODULO BIOS
0050 - 007F	RESERVADO BLOQUES CONTROL FICHEROS
0080 - 00FF	BUFFER DEL DISCO

Estructura interna en la memoria central del área SPA.



Estructura de la memoria primaria o central de un ordenador dotado del sistema operativo CP/M. (Las direcciones aparecen en código hexadecimal; los caracteres "XX" corresponden a direcciones genéricas).

zonas básicas mencionadas: zona para el sistema operativo (cuatro primeras porciones) y zona de los programas de aplicación (área TPA).

El cargador del sistema ("loader") posiciona los tres módulos del CP/M en la parte alta de la memoria, ocupando en conjunto un espacio de aproximadamente 6 Kbytes. El módulo BIOS (Basic Input Output System) es el localizado en el extremo superior. Tras éste se encuentra la zona destinada al módulo BDOS (Basic Disk Operating System) y, a continuación, la porción ocupada por el módulo CCP (Console Command Proccesor) que se referencia en su extremo inferior por medio del indicador CBASE.

Las referencias BIAS (afecta al extremo inferior de la memoria ocupada por el módulo BIOS) y CBASE son genéricas, ya que las direcciones hexadecimales de memoria que les corresponden son variables, al depender del espacio total de memoria interna que equipa al ordenador.

La porción que ocupa la zona baja de la memoria, con un total de 256 bytes, localizada entre las direcciones 0000 y 0100 (en expresión hexadecimal), también está destinada al sistema operativo y se denomina SPA (System Parameter Area). Su cometido es almacenar determinados parámetros del sistema y actuar como memoria auxiliar para operaciones propias de su misión.

La zona destinada a los programas de usuario ocupa, normalmente, el mayor espacio de memoria; está situada en la parte intermedia de la misma y se denomina TPA (Transient Program Area). Sus direcciones superior e inferior corresponden respectivamente con la referencia CBASE, que marca el extremo inferior de la zona de memoria correspondiente al módulo CCP, y la dirección 0100 (extremo superior del área de memoria SPA).

En algunos microordenadores —debido a su propia estructura de memoria y al hecho de existir ciertos programas pregrabados en memoria ROM— la dirección del extremo inferior del area TPA, coincidente con la superior del área SPA, se desplaza de la posición 0100 a la 4300.

Como se ha señalado, el área de memoria destinada a los programas de usuario es variable; depende de la capacidad global de memoria del ordenador. De cara a esta variación, la referencia inicial del área de memoria TPA es siempre fija (0100 ó 4300) y lo único que se altera es la dirección correspondiente a la referencia CBASE. Esta última variará de acuerdo a un múltiplo de 16 Kbytes (4000 en la referencia hexadecimal), desplazándose hacia el extremo superior de la memoria a medida que crece el volumen de la memoria central de la máquina.

Puede darse el caso de que un programa de aplicación resulte demasiado voluminoso para la capacidad del área de memoria destinada a almacenarlo (TPA). No obstante, ello no significa, en todos los casos, la imposibilidad de cargar el programa en memoria.

Cabe precisar que el sistema operativo CP/M puede permitir su carga, aunque con ciertas limitaciones. Al efecto, el sistema permite utilizar una porción del área de memoria destinada al módulo CCP, aunque no en su totalidad, y a su vez, procede a una redistribución de las zonas de memoria reservadas al conjunto del sistema operativo.

Lógicamente, el tamaño máximo del programa queda fijado por el área de memoria mínima con la que el sistema operativo es capaz de funcionar con eficacia. Una vez que se ha ejecutado el programa de aplicación, o bien debido a una intervención directa del usuario accionando la combinación de teclas <CTRL-C>, la memoria se reorganiza volviendo a la configuración inicial. Este proceso se denomina *"warm start"*.

Wordstar (y 3)

Una sesión de trabajo con el tratamiento de textos

Para finalizar el estudio de la aplicación para el tratamiento de textos WORDSTAR, vamos a simular una sencilla pero significativa sesión de trabajo. Evidentemente, no va a ser posible utilizar más que un pequeño subconjunto de las opciones disponibles; no obstante, ello sucede también así en la práctica totalidad de las sesiones reales.

INTRODUCCION

El trabajo con el WORDSTAR se encadena a través de una serie de menús, relacionados entre sí, que sirven de guía al operador. Para realizar la función deseada, éste se limitará a seleccionar cuál de las opciones ofrecidas en el menú es la oportuna. Aunque el WORDSTAR contiene una estructura de menús muy estudiada, cualquier usuario puede modificarla hasta acondicionarla a su gusto. El objetivo de la presente sesión de trabajo consistirá en producir un documento con el extracto de la situación económica de la empresa. Para ello será preciso seguir los pasos que se detallan en los próximos apartados.

APERTURA DEL DOCUMENTO Y ESCRITURA DEL TEXTO

Antes de empezar a trabajar es preciso comprobar que el disquete del WORDSTAR se encuentra disponible y, en caso afirmativo, será necesario invocar al sistema operativo CP/M. Tan pronto como aparezca en pantalla el indicador "A >", se tecleará "WS", terminando con una acción sobre la tecla <RETURN>. De esta forma tan simple, se logrará el arranque y puesta en escena del WORDSTAR; por supuesto, sin ningún archivo en edición.

En el primer menú, denominado menú sin archivo, es preciso utilizar la instrucción D para proceder a la apertura de un nuevo archivo. Tras ello, aparecerá en la pantalla el mensaje "ARCHIVO NUEVO" durante varios segundos. En ese instante puede ya afirmarse que nos encontramos en modo de *edición de archivos.*

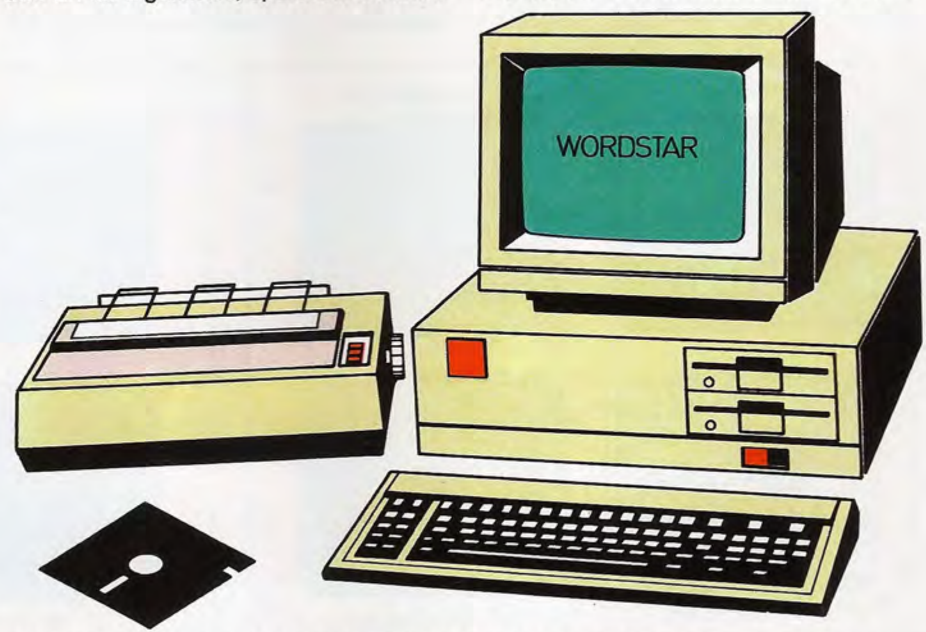
La zona superior de la pantalla se verá ocupada por una relación de las distintas posibilidades de edición. En la primera de estas líneas, denominada *línea de estado,* se puede observar el nombre del archivo (p. e. SITUACION.DOC), la página, la línea y columna en que se encuentra el cursor, etc. En las siguientes, aparecen descritas las funciones de edición, algunas de las cuales se utilizarán dentro de esta sesión práctica. A su vez, en la última línea se puede visualizar la tabulación de la siguiente forma:

L----!----!----!----!----!----!----!----R

en donde la L representa la columna de margen izquierdo, R la columna de margen derecho y las admiraciones indican la colocación de los tabuladores.

La parte de la pantalla que se encuentra situada debajo de la línea de tabulación, se denomina *área de exhibición del archivo.* En ella se muestra el texto del documento que se está editando; en nuestro ejemplo, como se trata de un archivo nuevo, el área de exhibición aparecerá en blanco.

Para escribir el texto: "La situación económica de la empresa es brillante, como se puede observar ..." basta, sencillamente, con teclear letra a letra los distin-



El tercer y último capítulo dedicado al WORDSTAR va a consistir en la simulación de una sesión de trabajo con el tratamiento de textos. Una actividad que, en la práctica, se realizará ante un sistema con una configuración semejante a la ilustrada.

tos caracteres. Si al escribir se sobrepasa el margen derecho, el WORDSTAR moverá, de forma automática, la última palabra a la siguientes línea; en consecuencia, el operador no tendrá que preocuparse de la terminación de líneas en el texto. A este proceso se le denomina *fin de línea automático*.

Otra propiedad importante es la denominada *alineación* que consiste en la inserción de espacios entre las palabras, de tal forma que todas las líneas finalicen en la columna marcada con una R en la tabulación.

Al escribir el texto, la tecla <RETURN> tan sólo se pulsará cuando se dé por terminado un párrafo.

MODIFICACIONES DEL TEXTO ORIGINAL

Una vez tecleado el texto completo, puede surgir la necesidad de realizar modificaciones de distinto tipo en el mismo. Si dicha necesidad se detecta antes de concluir la edición inicial, se pueden realizar las modificaciones inmediatamente; no obstante, si el fichero se encuentra ya almacenado, será necesario editarlo para realizar las correcciones. Las principales opciones de modificación son las siguientes:

- *Movimiento del cursor*

El primer paso para realizar modificaciones en un documento ya existente, es situar el cursor en la posición que se desee actualizar. Para ello se pueden utilizar las instrucciones ^S, ^D, ^A, ^F, ^E y ^X que, respectivamente, significan ir a la letra izquierda, ir a la letra derecha, ir a la palabra izquierda, ir a la palabra derecha, ir a la línea superior e ir a la línea inferior.

Si en nuestro ejemplo el cursor se encuentra situado en la columna 1 de la línea 1, y deseamos modificar el texto en la columna 6 y la línea 10, habrá que pulsar ^D cinco veces y ^X nueve veces, hasta situar el cursor en el lugar deseado. Si se quieren realizar desplazamientos rápidos a través de la línea, es preferible usar las instrucciones ^A y ^F; éstas desencadenan saltos de palabra en palabra (sea cual fuere su longitud), en lugar de desplazarse carácter a carácter.

Una vez situado el cursor en la posición apropiada, al escribir nuevos caracteres éstos se insertarán "empujando" a los antiguos. A este proceso se le denomina *inserción* y se produce por defecto. Si se desea modificar —no insertar, sino reescribir—, los nuevos caracteres tecleados sustituirán a los antiguos, sin que se produzcan desplazamientos en el texto.

En la línea de estado, se puede visualizar si la edición está en modo inserción o no. Para entrar en modo inserción basta, sencillamente, con utilizar la instrucción ^V.

- *Borrado de caracteres*

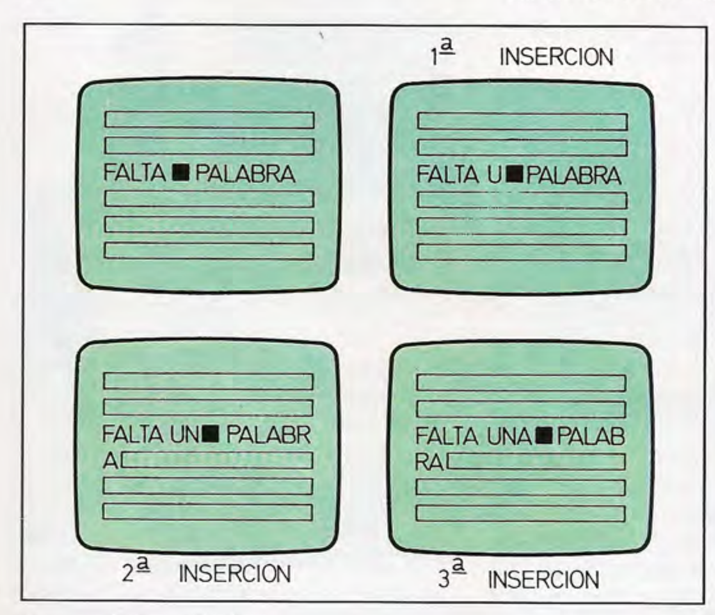
Tras efectuar una corrección, es posible que queden algunos caracteres sobrantes a la derecha del cursor. Para borrarlos, hay que utilizar la instrucción ^G (borrar el carácter de la derecha). Al contrario del "empuje" de caracteres producido en el modo inserción, el borrado de un carácter implica la "absorción" de un espacio, de forma que el resto de la línea se desplaza una posición hacia la izquierda.

Después de haber realizado modificaciones en un texto, el margen derecho puede que ya no esté alineado; es responsabilidad del operador realizar los ajustes necesarios, ya que el WORDSTAR no se encarga en este caso de romper líneas demasiado largas.

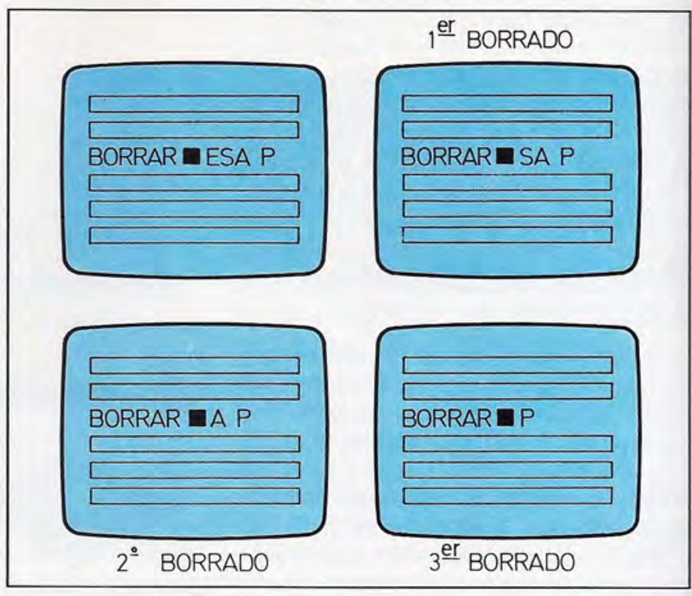
- *Realineación del margen derecho*

La instrucción ^B sirve para *realinear* un párrafo que fue escrito con fin de línea automático. Para ello, se coloca el cursor encima de la primera letra alterada antes de ordenar que se ejecute la realineación. De esta forma se modificarán todas las líneas del párrafo, esto es: desde la posición del cursor hasta la última vez que se pulsó <RETURN> en la introducción del texto.

Si se utiliza la realineación, al encontrar una palabra demasiado larga en un final de línea, el WORDSTAR puede proponer al operador su separación mediante un guión; al efecto muestra el oportuno mensaje y detiene la realineación. Con



Proceso de inserción de caracteres en un texto en edición.



La instrucción G permite el borrado de caracteres con recuperación de espacio, tal como muestra la secuencia de pantallas.

ello, permite al usuario que teclee el carácter "–", o bien que oprima ^B y continúe la realineación sin colocar el guión propuesto. El proceso de realineación es relativamente lento, hasta el punto de que si se trabaja con un párrafo largo, el tiempo invertido por el WORDSTAR puede rozar los 30 segundos. Para que el operador sepa en todo momento el estado del procesador, en la esquina superior izquierda de la pantalla aparecerá el indicativo "^B"; tan sólo después de finalizar completamente la realineación, o cuando se proponga una separación de palabra, se podrá observar el efecto producido.

- *Inserción de líneas*

En determinados casos puede ser necesario introducir párrafos completos entre párrafos ya escritos. Por ejemplo, en nuestro informe de situación económica ya introducido, es posible que se desee añadir un nuevo párrafo, indicando un factor de valoración no considerado inicialmente. Para ello se puede utilizar la instrucción ^N: cada vez que se introduzca aparecerá una nueva línea en blanco debajo de la que contenga el cursor.

- *Scrolling*

Cuando la pantalla queda repleta de líneas, ya sea en la carga inicial, o bien en una modificación, el texto empezará a desplazarse hacia arriba cada vez que el cursor salte a una nueva línea. Si el operador es muy rápido tecleando, puede darse el caso de que su velocidad sea superior a la de aparición de los caracteres en la pantalla; no obstante, la pantalla se actualizará cuando "pueda", sin necesidad de que el operador tenga que detener su trabajo.

En algunos casos aparecerá la palabra "ESPERE" en la línea de estado, a la vez que se oirá una señal de aviso destinada al operador. Cuando esto ocurre, es recomendable dejar de escribir o escribir lentamente, hasta que el WORDSTAR se recupere y desaparezca el mensaje.

- *Fin de página*

Cuando se introduce suficiente texto como para llenar una página del informe, aparecerá una línea de guiones a lo largo de la pantalla con una "P" en la última posición:

"-------------------- . . .--------------------P".

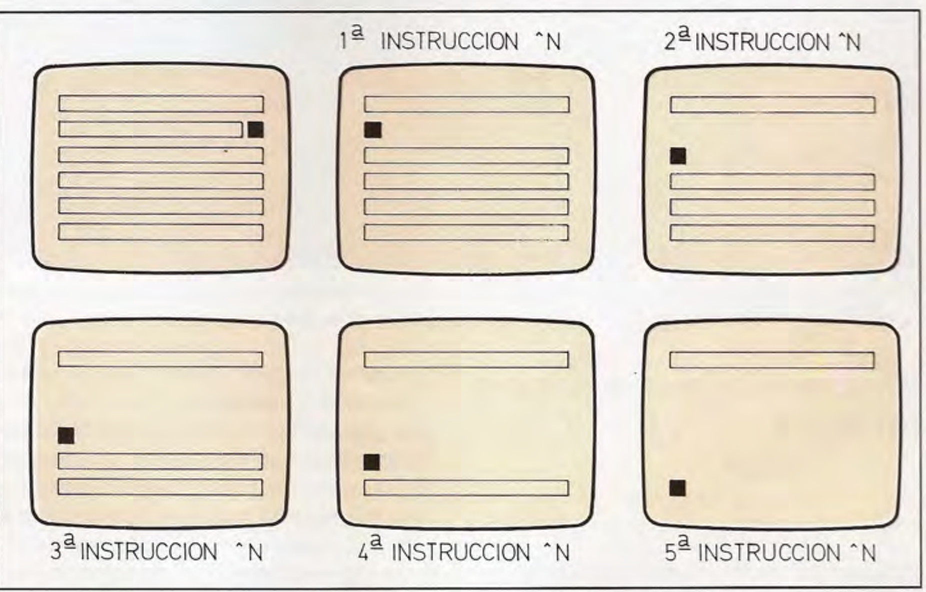
INSTRUCCIONES PARA MOVER EL CURSOR, RECORRER EL TEXTO O REALIZAR BUSQUEDAS	
^W	Recorre abajo una línea.
^E	Desplazamiento a la línea superior.
^R	Salto a la pantalla anterior.
^A	Salto a la palabra de la izquierda.
^S	Salto al carácter de la izquierda.
^D	Salto al carácter de la derecha.
^F	Desplazamiento a la palabra de la derecha.
^Z	Recorre hacia arriba una línea.
^X	Salto a la línea inferior.
^C	Desplazamiento a la siguiente pantalla.
^Q^W	Recorre hacia abajo continuamente.
^Q^E	Salto a la parte superior del área de texto.
^Q^R	Salto al principio del archivo.
^Q^A	Sustituye texto.
^Q^S	Desplazamiento al lado izquierdo de pantalla.
^Q^D	Desplazamiento al lado derecho de pantalla.
^Q^F	Busca texto.
^Q^Z	Recorre hacia arriba continuamente.
^Q^X	Cursor a la parte inferior área de texto.
^Q^C	Cursor al final del archivo.
^QO-9	Cursor a la marca.
^Q^P	Cursor a la posición anterior a la última instrucción.
^Q^K	Cursor al final del bloque.
^Q^V	Cursor al punto de partida del último busca u origen del último bloque.
^Q^B	Cursor al principio del bloque.

Esta es la indicación dinámica de fin de página. Si se está de acuerdo con el punto en el que ocurre, el usuario debe seguir escribiendo normalmente. En caso contrario, puede recurrir a determinadas instrucciones para especificar dónde deben finalizar las páginas, e incluso detallar cuestiones de formato (por ejemplo, indicar el número de líneas que deben escribirse en cada página). A este tipo de instrucciones se las denomina "de punto", dado que sintácticamente todas ellas comienzan por el carácter " . "; por ejemplo:

.PL sirve para especificar la longitud del papel

.MT para definir el tamaño del margen superior

.MB pare definir el margen inferior ...



Secuencia de inserción de nuevas líneas en blanco, por medio de la instrucción N, para la introducción de un párrafo en el texto original.

RESUMEN DE INSTRUCCIONES DE TIPO "PUNTO" DEL WORDSTAR			
Instruc.	Función	Unidades	Inicialmente
.LH	Altura de líneas	1/48 pulgadas	8=6 líneas por pulgada
.PL	Longitud Papel	líneas	66 líneas=11 pulgadas
.MT	Margen Superior	líneas	3 líneas=1/2 pulgada
.MB	Margen Inferior	líneas	8 líneas=1 1/3 pulgadas
.HM	Margen Encabezado	líneas	2 líneas=1/3 pulgada
.FM	Margen Pie página (margen £ de página)	líneas	2 líneas=1/3 pulgada
.PC	Columna £ de página	columnas	1/2 margen derecho inicial
.PO	Corrimiento página	columnas	8 columnas=4/5 pulgada
.PA	Cambio de página		
.CP	Página Condicional	líneas	
.HE	Encabezado		vacío
.FO	Pie de página		número de pág. en columna .PC
.OP	Omite números de página		
.PN	Número de página		1
.CW	Ancho de Caracteres	1/120 pulgadas	12 para ancho estándar 10 para ancho alternativo
.SR	Baja Subíndices	1/48 pulgadas	3
.UJ	Microalineación	NO(0)SI(1)	SI (1)
.BP	Impresión bidireccional	NO(0)SI(1)	SI (1)
.IG	Comentario (o ...)		



El repertorio de comandos para la edición en pantalla del WORDSTAR, sintetiza las funciones necesarias para automatizar y dar eficacia a la producción de textos.

REVISION Y ALMACENAMIENTO DEL TEXTO

Para revisar todo el texto introducido antes de almacenarlo en disquetes o imprimirlo en papel, pueden utilizarse las instrucciones de subir o bajar "línea a línea" o "pantalla a pantalla".

Una vez satisfechos con el contenido del archivo en edición se puede, y debe, proceder a su almacenamiento. Todas las operaciones realizadas en la sesión de trabajo sólo tienen efecto transitorio, de ahí que si no se transfiere el documento a un archivo, no será posible conservar todo lo escrito.

La instrucción básica de almacenamiento es ^KD; ésta se encarga de grabar toda la información en un archivo en disco, con el nombre elegido al empezar la edición.

IMPRESION DE UN DOCUMENTO

El fin último de todo documento editado (y nuestro informe económico no va a ser una excepción) no es otro que obtener su impresión en papel. Para ello, el operador debe asegurarse inicialmente que la impresora está preparada (encendida, con el papel colocado, ...). Acto seguido, introducirá la opción P del menú sin archivo, a lo que el WORDSTAR reaccionará con una pregunta:

¿NOMBRE DEL ARCHIVO QUE SE DESEA IMPRIMIR?

En ese momento, el operador debe contestar con el nombre del archivo y, a continuación, pulsar la tecla <RETURN>; sucesivamente, irán apareciendo nuevas preguntas que conducirán a la obtención del documento escrito a través de la impresora.

Mientras la impresora está escribiendo un documento, es posible editar otro distinto y seguir trabajando. Sin embargo, en estas condiciones, el tiempo de respuesta será muy superior; de ahí que es recomendable utilizar sólo edición simultánea para revisiones de textos y/o modificaciones no excesivamente importantes.

RECAPITULACION

En esta hipotética sesión de trabajo con el WORDSTAR, se han puesto en juego una mínima parte de las instrucciones que brinda la aplicación. Desde luego, la descripción completa de una sesión de trabajo real exigiría mucho más espacio y no es el objetivo de la obra. Por ello, la descripción se limita a los pasos usuales e imprescindibles para procesar un documento sencillo.

Almacenamiento de programas

Grabación y lectura de programas en la memoria auxiliar

Son ya numerosos los comandos y las funciones BASIC que han desfilado en estas páginas de la obra. A estas alturas, se han volcado ya conocimientos suficientes para permitir la confección de programas, no excesivamente complicados, aunque no por ello menos interesantes.

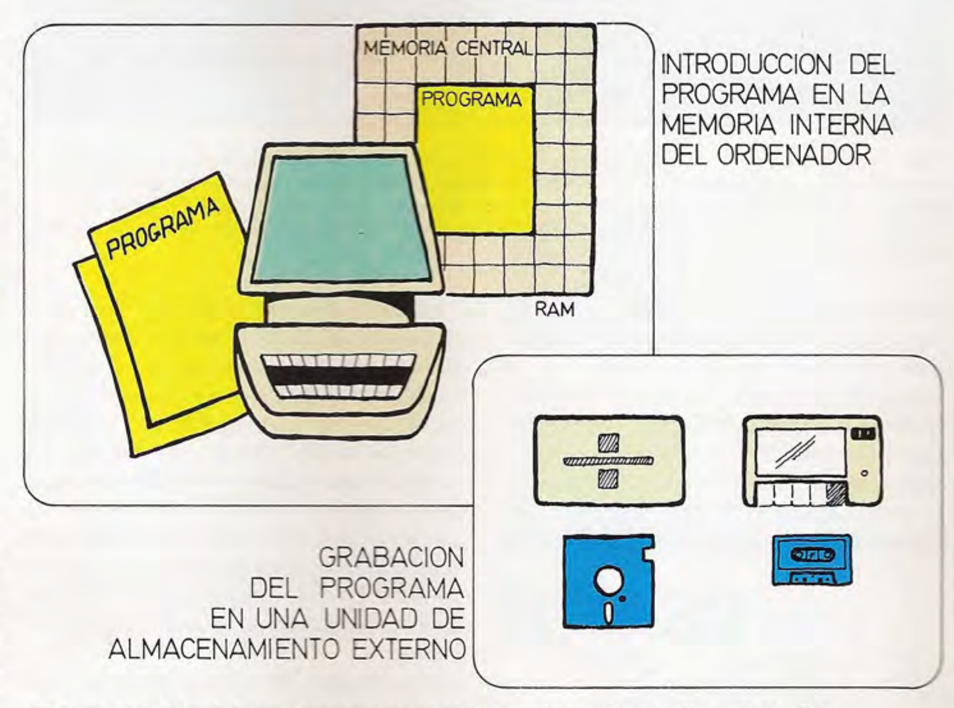
¿COMO CONSERVAR LOS PROGRAMAS?

Un hecho fácilmente constatable es que una vez que se desconecta la alimentación del ordenador, la memoria interna de éste se borra y, en consecuencia, el programa que reside en su interior desaparece. Ello constituye un grave problema, ya que las horas de trabajo invertidas en la redacción de un programa se pierden, simplemente, al desconectar el aparato. No hay que olvidar que tal desconexión puede producirse de una forma involuntaria: por fallo en la alimentación, o al cometer un error en la manipulación del aparato.

El objetivo habitual de un programa no se reduce a ejecutarlo tan sólo una vez. De ellos se espera que sirvan para resolver una necesidad o realizar una acción más o menos frecuente. Si el operario o programador tuviese que confeccionar de nuevo el programa, cada vez que surja la necesidad de resolver un problema con el ordenador, contaría con un motivo más que suficiente para disuadirle de emplear este método de trabajo; posiblemente buscaría y encontraría métodos más cómodos y eficaces.

No cabe duda que ésta no es una situación envidiable. En cualquier caso, cabe la opción de tomar nota del programa y, más tarde, cuando sea necesario, introducirlo de nuevo en la memoria del ordenador procediendo a su escritura desde el teclado. Tampoco ésta es una solución idónea. El método sigue resultando incómodo y poco eficaz. La reiterada escritura del programa siempre es una operación tediosa y que, por lo general, suele acarrear problemas. Por ejemplo, un error cometido al teclearlo hará que el programa no funcione correctamente; será necesario localizar los errores cometidos en una labor complicada y que no siempre conduce a un resultado inmediato.

Pero no hay por qué alarmarse; éstas no son las alternativas habituales. El ordenador pone a disposición del usuario otras



El objetivo habitual de un programa no se reduce a ejecutarlo tan sólo una vez. En consecuencia, tras introducirlo en el ordenador a través del teclado, es conveniente grabarlo en un soporte de memoria de tipo permanente (por ejemplo, cinta en casete o disco).



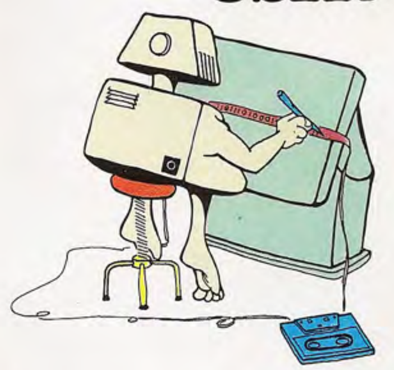
Las casetes y, en general, las cintas magnéticas son soportes de memoria de tipo secuencial. En ellas la información se graba en serie, dato a dato. De ahí que para acceder a un dato específico, haya que pasar por todos los almacenados en posiciones precedentes.



Las unidades de casete constituyen el periférico de almacenamiento externo más económico. Su presencia es frecuente junto a los ordenadores personales de tipo doméstico. Algunos equipos deben utilizar una unidad de casete específicamente diseñada por el fabricante; no obstante, otros equipos son capaces de trabajar asociados a cualquier magnetófono de audio de tipo común.

técnicas bastante más apropiadas; técnicas que permiten almacenar los programas para su posterior reutilización en el momento adecuado. De esta forma, para conservar el programa, será suficiente con ordenar a la máquina que lo transfiera a un soporte de memoria permanente, en el que la información almacenada no desaparezca al desconectar la alimentación.

CSAVE



Cada vez que sea preciso utilizar de nuevo el programa, bastará sencillamente con que el ordenador lo extraiga de la memoria permanente y lo deposite en su memoria interna. Los dispositivos externos encargados de realizar una función de tanta utilidad reciben el nombre de unidades de almacenamiento masivo o memorias auxiliares. Mediante la oportuna codificación, es posible almacenar en ellas la información que, una vez leída por el ordenador, le permitirá reconstruir el programa.
Las unidades de almacenamiento más empleadas con los ordenadores personales son las unidades de disco o de casete. Ambas unidades comparten una característica común: basan su funcionamiento en las propiedades magnéticas de ciertos materiales. Sin embargo, existen notables diferencias entre ellas, y no sólo por lo que respecta a su constitución física, sino también en cuanto a su funcionamiento. En principio, las unidades de disco son notablemente más rápidas que las de casete; a su vez, en las primeras es posible acceder directamente a cualquier punto de la superficie del disco, con lo cual, se puede leer cualquiera de los programas almacenados en el disco sin por ello, pasar obligatoriamente por la lectura de los emplazados en posiciones previas. Desgraciadamente, ello no es posible en las unidades de casete. Estas almacenan la información secuencialmente, dato a dato, en la superficie de una cinta de tipo casete. Una idea elocuente se obtiene al pensar en los casetes y discos de audio, con los que sucede algo semejante. En un disco se puede oír cualquier canción con sólo posicionar la cabeza lectora en el lugar adecuado de su superficie, mientras que en el casete la cosa se complica, ya que es preciso bobinar o rebobinar la cinta hasta situarla en el lugar adecuado, en consecuencia, hay que pasar por encima de otra información que no reviste interés en el acceso actual.
En los próximos apartados se describirá la forma en la que se almacenan los programas en las unidades de almacenamiento más usuales, añadiendo algunas notas prácticas relativas a su empleo.

GRABACION DE PROGRAMAS EN CASETE

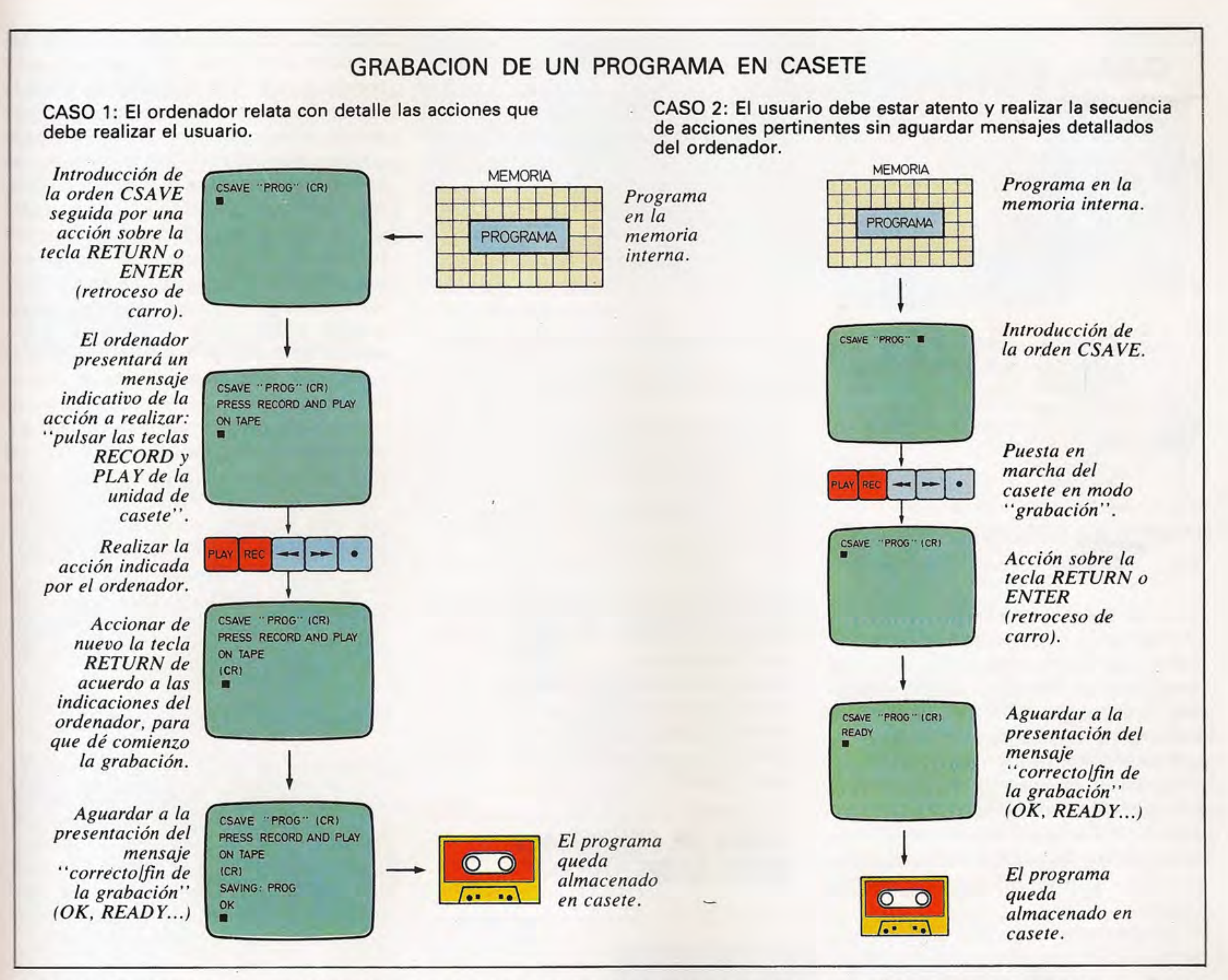
Las unidades de casete constituyen una de las memorias de masa más utilizadas en equipos domésticos. El motivo fundamental radica en su economía. La simplicidad de estas unidades hace que resulten fáciles de construir y muy robustas. En muchos casos, incluso, no será necesario adquirir una de estas unidades, ya que su función puede realizarla un simple magnetófono para casetes de audio de tipo convencional. Ello aporta algunas ventajas. Raro es el aficionado que no dispone de un magnetófono a casete, de ahí que el uso de estos periféricos para el almacenamiento masivo resulte muy asequible y, desde luego, económico.
También los soportes empleados, las cintas en casete, son baratas y fáciles de adquirir. En principio, cualquier cinta comercial de audio resulta adecuada, si bien, es conveniente huir de las cintas de escasa calidad, ya que pueden ocasionar problemas que conducirán a la pérdida de la información en ellas almacenada. Para

CSAVE

Almacena un programa en la unidad de casete, otorgando al mismo el nombre especificado.

Formato: CSAVE "<nombre del programa>"

Ejemplo: CSAVE "CASA"



empezar el trabajo con la unidad de casete, hay que contar con un programa, por ejemplo el siguiente:

```
10 REM PROGRAMA DE PRUEBA
20 PRINT "PROGRAMA"
30 PRINT "DE DEMOSTRACION"
40 PRINT "PARA EL ALMACENAMIENTO"
50 PRINT "DE INFORMACION"
60 PRINT "EN CASETE"
■
```

Acto seguido, será preciso conectar la referida unidad al ordenador; para ello, es necesario consultar el manual de cada equipo, ya que en este punto no caben métodos generales.

Ahora puede ya realizarse la grabación del programa en casete. Para ello, es necesario que el ordenador transfiera una señal a la unidad de casete, señal que será grabada para su posterior uso. Desde luego, la transferencia no se va a realizar de una forma automática, sólo con que el usuario lo desee, es necesario comunicar la orden adecuada al equipo. Esta coincide, en muchos casos, con el comando BASIC CSAVE, cuya formulación es la siguiente:

CSAVE "<nombre del fichero>"

El nombre del fichero es una referencia que identificará al fichero o bloque de datos (el programa en este caso) y que facilitará su manipulación. No hay que perder de vista que un mismo casete suele constituir el soporte de almacenamiento de más de un programa; por lo tanto, el nombre del fichero permitirá identificar a cada uno de los programas por separado. Dependiendo del ordenador del que se trate, el nombre debe cumplir determinadas condiciones; por lo general, éstas se refieren a la longitud máxima del mismo, habitualmente de seis a ocho caracteres. Por ejemplo:

CSAVE "CUENTA"
CSAVE "PROG"

En ciertos casos, es posible añadir un apellido al nombre del archivo. Este apellido es una extensión que acompaña al nombre; consiste en dos o tres caracteres, situados a la derecha del nombre, y separados de éste por un punto.

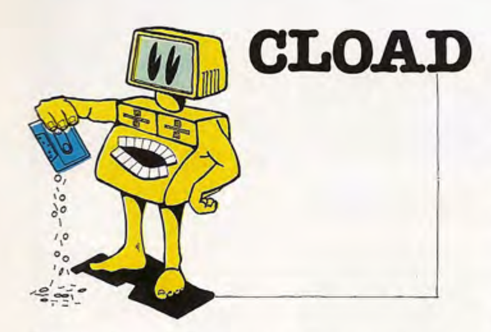
Al ejecutar una instrucción CSAVE, el ordenador transferirá el programa que se encuentra en la memoria al casete, en

CLOAD

Carga en la memoria interna del ordenador un programa almacenado en casete.

Formato: CLOAD "[<*nombre del programa*>]" [, *R*]

Ejemplos: CLOAD "PROG"
CLOAD " "
CLOAD "TEMA", R



forma de señal eléctrica. Esta contiene la información necesaria para que, posteriormente, sea posible realizar el proceso inverso, esto es: leer dicha señal y reconstruir el programa en el ordenador. Para ello será necesario que la unidad de casete se encuentre en condiciones de recibir la señal y grabarla. Al respecto, habrá que pulsar simultáneamente las teclas de grabación y avance del magnetófono. Desde luego, es preciso sincronizar esta operación con la introducción del comando.
En algunas ocasiones, esta tarea es bastante sencilla; una vez introducido el comando, el ordenador pedirá al usuario que ponga a la unidad de casete en situación y, tras ello, que accione una tecla (RETURN, normalmente) para que dé comienzo el proceso de grabación. En las correspondientes figuras aparecen dos secuencias de operación propias de distintos ordenadores.
El proceso de grabación no tiene una duración fija, sino que depende de la longitud del programa y también de la velocidad con la que el ordenador es capaz de enviar los datos a la unidad. El proceso puede durar desde unos segundos hasta varios minutos. En cualquier caso, no hay por qué preocuparse; el ordenador se encargará de informar al operario en el instante en el que termine la grabación, presentando en la pantalla el oportuno mensaje. Tras ello, es conveniente detener el casete para que no siga grabando; en algunos casos —si se trata de un casete especial— el propio ordenador se encargará de detenerlo.
Algunos ordenadores no permiten la grabación de un programa con nombre; de ocurrir así, será suficiente con introducir el comando de grabación sin argumento alguno: CSAVE.

CARGA DE PROGRAMAS DESDE CASETE

Cualquier programa grabado en cinta magnética, ya sea por el usuario mediante el comando CSAVE, o bien grabado por el fabricante del casete si se trata de un programa comercial, puede cargarse en el ordenador y ejecutarse en cualquier momento.
La necesidad del almacenamiento de los programas es una realidad apuntada en párrafos anteriores. En cualquier momento puede desconectarse accidentalmente la alimentación del aparato, o el propio usuario, por error, puede introducir un comando NEW. ¿Qué sucede en tal caso? La respuesta es que la memoria del ordenador quedará "limpia". La ejecución de un comando LIST no obtendrá respuesta; lo mismo sucederá con el comando RUN. El programa se ha perdido. No obstante, si se tomó la precaución de grabarlo previamente, será posible recuperarlo en cualquier instante. De ahí que cuando se esté desarrollando un nuevo programa, resulte conveniente tomarse la molestia de hacer copias de seguridad en las sucesivas versiones, a medida que se vaya perfeccionando el mismo. De no hacerlo así, el usuario puede llevarse alguna que otra sorpresa, especialmente en lo que se refiere a la alimentación del aparato, siempre a expensas de la red.
El comando adecuado para ordenar la carga de un programa almacenado en cinta es, normalmente, CLOAD. Su aspecto general es el siguiente:

CLOAD "<nombre del programa>"

El nombre del programa ha de coincidir con el utilizado en el proceso de grabación. Sin embargo, si no se recuerda el nombre, éste puede omitirse; en tal caso, el ordenador cargará el primer programa que localice en la casete. A continuación, se muestran algunos ejemplos de formulaciones válidas del comando CLOAD:

```
CLOAD "PEPE"
CLOAD "NOMBRE"
CLOAD " "
```

La forma de trabajar con este comando es muy simple. Una vez introducido a través del teclado y tras accionar la tecla RETURN (retorno de carro), el ordenador pasará a examinar las señales procedentes de la unidad de casete. En el caso de que la señal analizada coincida con el programa que se desea cargar, procederá a su lectura y a su transferencia a la memoria interna. Para evitar errores durante la operación es conveniente situar la cinta lo más cerca posible del punto en el que comienza el programa. Para ello se hace necesario disponer de alguna referencia que permita identificar el punto en cuestión; por ejemplo, bastará con apoyarse en el estado del cuentavueltas que acos-

CLOAD?

Verifica la coincidencia de un programa grabado en cinta con el que se encuentra en la memoria interna.

Formato: CLOAD? "<*nombre del programa*>"

Ejemplo: CLOAD? "PEP"

tumbra a equipar a las unidades de casete.

Como ya se indicó, algunos ordenadores no admiten el campo correspondiente al nombre dentro del comando CSAVE. Por supuesto, tampoco lo admitirán ahora como argumento de CLOAD. En tal caso, la carga se ordenará de forma análoga a cuando se desconoce el nombre otorgado al fichero: CLOAD" ".

VERIFICACION DE LOS PROGRAMAS ALMACENADOS

Las unidades de casete no están exentas de problemas; unas veces porque falla la alimentación en un cierto instante, otras porque las cabezas lectoras o grabadoras están sucias, e incluso, en ocasiones, por efecto de interferencias originadas por señales de radio o por algún aparato eléctrico conectado a la red de tensión. Todas estas circunstancias repercuten en que la calidad de las grabaciones sea, a veces, defectuosa, hasta el punto de que resulte imposible recuperar el programa grabado.

No cabe duda que es conveniente tomar algunas medidas precautorias. La primera de ellas consiste en realizar varias grabaciones de un mismo programa, en la misma cinta o, preferiblemente, en otra distinta. A pesar de todas las precauciones, siempre queda la incertidumbre de saber si el programa se grabó correctamente.

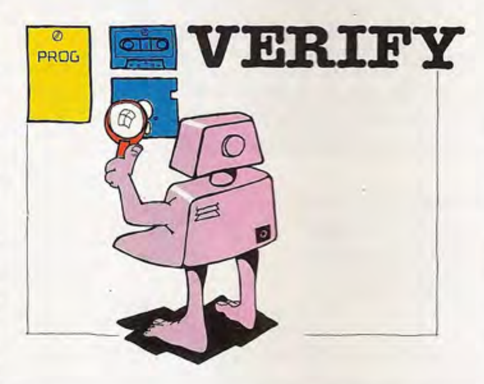
El método más inmediato para eliminar tal incertidumbre consiste, sencillamente, en borrar la memoria por medio del comando NEW y proceder a la carga inmediata del programa, para verificar si es posible recuperarlo. Desde luego, la operación eliminará cualquier incertidumbre, pero poco más, puesto que el error ya no será reparable. Si la grabación ha sido defectuosa, se habrá perdido el programa irremisiblemente. Muchos dialectos BASIC incorporan un comando especializado que permite comprobar si la grabación se ha efectuado correctamente. Este suele ser el comando CLOAD?, cuyo formato es:

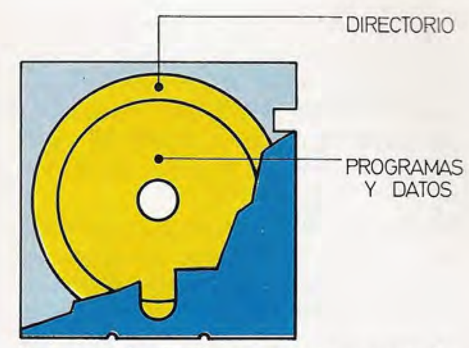
CLOAD? "<nombre del programa>"

En efecto, el nombre del programa debe coincidir con el utilizado en el instante de su grabación.

Veamos cuál es su actuación. Una vez que se ha grabado el programa con la orden CSAVE, y antes de borrar la memoria o de introducir cualquier modificación en el programa, puede entrar en escena el comando de verificación. En primer lugar, hay que rebobinar la cinta hasta el punto en el que dio comienzo la grabación; tras ello, se introducirá la orden CLOAD? acompañada por el nombre con el que fue grabado el programa a verificar. La ejecución del mismo hará que el ordenador compruebe la total coincidencia del programa que reside en su memoria con el programa del mismo nombre que se encuentra en la casete.

En algunos dialectos BASIC este comando no recibe el nombre de CLOAD?, sino que obedece al nombre de VERIFY; no obstante, su funcionamiento es análogo al descrito.





Las unidades de disco permiten conocer, de forma casi instantánea, cuál es el número y nombre de los programas almacenados en cada disco magnético. Ello es posible debido a que las referidas unidades reservan una zona del disco, denominada "directorio", para el almacenamiento de dicha información.

ALMACENAMIENTO EN UNIDAD DE DISCO

Hasta ahora se ha hablado exclusivamente de comandos relacionados con la grabación y recuperación de programas en unidades de casete. Esta situación no deja de ser frecuente, sin embargo está lejos de ser la más eficaz. Existen otros periféricos de almacenamiento, también popularizados, capaces de prestar un mejor servicio.



La velocidad de transferencia de información en ambos sentidos (grabación y lectura) no es una de las características propias del casete. Cuando las exigencias se concretan en la necesidad de almacenar un gran volumen de información, y garantizar un acceso rápido y directo a la misma, es preciso optar por el almacenamiento en disco.

Al margen de las unidades de casete, las más extendidas son las de disco flexible. Estas últimas presentan grandes ventajas, especialmente por lo que respecta al tiempo de acceso a los programas y a la velocidad de carga y grabación de los mismos.

Una notable prestación de las unidades de disco es que, de manera instantánea, es posible conocer el número de programas almacenados en el disco, así como sus respectivos nombres. Hay que tener en cuenta que las unidades de disco reservan una zona del soporte, denominada *directorio,* para almacenar dicha información. El directorio es una suerte de índice o catálogo en el que están reflejados todos y cada uno de los programas que se han ido grabando, señalando su localización exacta en el soporte. Con ello, es posible acceder directamente a cada uno de los programas tras una consulta al directorio.

La desventaja de estas unidades frente a las de casete reside en el precio, notablemente superior; aunque como contrapartida ofrecen unas prestaciones superiores, que permiten el empleo de pequeños ordenadores en aplicaciones (comerciales, de gestión...) que sin su colaboración quedarían vedadas.

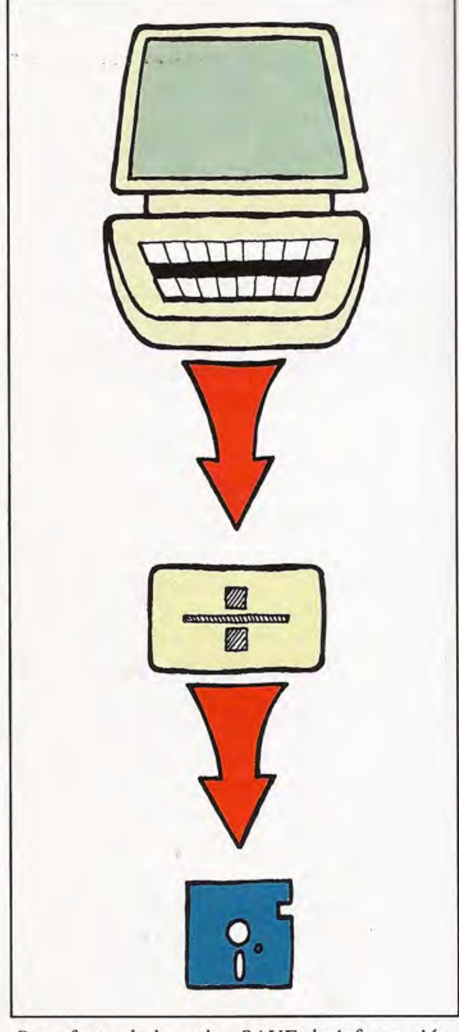
La operación básica a realizar con una unidad de disco consiste en la grabación de programas. El comando BASIC al efecto suele coincidir con SAVE, cuya formulación habitual es:

SAVE "<periférico> : <nombre del programa>"

Realmente, SAVE tiene en muchos ordenadores la doble utilidad de ordenar el almacenamiento de información tanto en casete como en disco. Ese es, precisamente, el motivo por el que se incluye la opción <periférico> en su argumento; ésta debe referenciar a la unidad en la que hay que grabar la información. La opción a utilizar para que el programa sea enviado a la unidad de casete es CAS, aunque, normalmente, también será éste el destino del programa si no se especifica referencia alguna. Para elegir la unidad de disco como destinataria de la grabación hay que colocar en dicho campo el número de la unidad implicada (1, 2, ...). El nombre del programa ha de ajustarse a unas condiciones semejantes a las que se imponían en el caso del casete; esto es: no debe sobrepasar una determinada longitud (longitud que varía de uno a otro ordenador) y puede admitir o no "apellidos", dependiendo de la máquina en cuestión. Los siguientes ejemplos son formulaciones habituales del comando SAVE:

```
SAVE "CAS.LOC1"
SAVE "1:GUSA"
SAVE "2:CASA"
SAVE "PROG"
```

Cuando se trabaja con una unidad de casete, el usuario se ve obligado normalmente a realizar las funciones de controlador de la cinta; en tal caso, el ordenador trabaja "a ciegas" grabando o leyendo de la zona en la que esté posicionado el cabezal, y sin comprobar si en ese lugar había algo previamente grabado. Desde luego, ello significa que en una misma cinta es posible grabar dos programas con el mismo nombre. Esta característica no es extensiva a la unidad de disco. Ahora, el ordenador es el encargado de controlar el disco, auxiliándose para ello en el directorio. En consecuencia, si se ordena la escritura de un programa cuyo nombre coincide con el de otro programa presente en el disco, el ordenador grabará dicho programa encima del ya existente. Dado que el control de la unidad de disco es automático, el trabajo del operador se limita a comunicar a la máquina las instrucciones adecuadas.



Por efecto de la orden SAVE, la información contenida en la memoria central del ordenador fluye hacia la unidad de almacenamiento externo; esta última se encarga de grabarla en el soporte adecuado.

SAVE

Graba el programa que se encuentra en la memoria del ordenador en una unidad de almacenamiento externo (disco o casete).

Formato: SAVE "[<*periférico de destino*>:] [<*nombre del programa*>]"

Ejemplos: SAVE "CAS : GAME"
SAVE "1 : CATO"

LECTURA DE PROGRAMAS EN DISCO

Para recuperar los programas de la unidad de disco es necesario recurrir al comando

TABLA DE CONVERSION

ORDENADOR	CLOAD	CSAVE	Verificación	LOAD	SAVE	Autoejecución	
	CLOAD "<nom.>"	**CSAVE "<nom.>"**	**CLOAD? "<nom.>"**	**LOAD "<per.>: <nom.>"**	**SAVE "<per.>: <nom.>"**	**CSAVE "<nom.>", AUTO**	**LOAD "<per.>: <nom.>", R**
APPLE II (APPLESOFT)	—	—	VERIFY "[nom.]"	LOAD "[nom.]"	SAVE "[nom.]"	—	RUN [nom.]
APRICOT (M-BASIC)	—	—	—	LOAD "<nom.>"	SAVE "<nom.>"	—	LOAD "<per.>: <nom.>", R RUN "<nom.>" [<, R>]
ATARI	CLOAD	CSAVE	—	LOAD "<per.>: <nom.>"	SAVE "<per.>: <nom.>"	—	—
CBM 64	—	—	VERIFY ["<nom.>"] [, <per.>]	LOAD ["<nom.>"] [, <per.>]	SAVE ["<nom.>"] [, <per.>]	—	(1)
DRAGON	CLOAD "<nom.>"	CSAVE "<nom.>"	—	—	—	—	—
EQUIPOS MSX	CLOAD ["<nom.>"]	CSAVE "<nom.>"	CLOAD? ["<nom.>"]	LOAD "<per.>: <nom.>"	SAVE "<per.>: <nom.>"	—	LOAD "<per.>: <nom.>", R
HP-150	—	—	—	LOAD "<nom.>"	SAVE "<nom.>"	—	LOAD "<nom.>", R
IBM PC	—	—	—	LOAD "<per.>: <nom.>"	SAVE "<per.>: <nom.>"	—	LOAD "<per.>: <nom.>", R
MPF	LOADT "<nom.>"	SAVET "<nom.>"	—	—	—	—	—
NCR DM-V (MS-BASIC)	CLOAD "<nom.>"	CSAVE "<nom.>"	CLOAD? "<nom.>"	LOAD "<nom.>"	SAVE "<nom.>"	—	LOAD "<per.>: <nom.>", R
NEW BRAIN	LOAD	SAVE "<nom.>"	VERIFY	—	—	—	—
ORIC	CLOAD "[<nom.>]"	CSAVE "<nom.>"	CLOAD "[<nom.>]", V	—	—	CSAVE "<nom.>", AUTO	—
SHARP MZ-700 (MZ-BASIC)	LOAD "[<nom.>]"	SAVE "[<nom.>]"	VERIFY "[<nom.>]"	—	—	—	—
SINCLAIR QL	CLOAD "<nom.>"	—	—	LOAD ["] <per.> <nom.> ["]	SAVE ["] <per.> <nom.> ["]	—	—
SPECTRA-VIDEO	CLOAD "<nom.>"	CSAVE "<nom.>"	CLOAD? "<nom.>"	LOAD "<per.>: <nom.>"	SAVE "<per.>: <nom.>"	—	LOAD "<per.>: <nom.>", R
ZX-SPECTRUM	LOAD "[<nom.>]"	SAVE "[<nom.>]"	VERIFY "[<nom.>]"	—	—	SAVE "[<nom.>]" LINE <n>	—

<nom.>: Nombre del programa con el que se trabaja. <per.>: Especifica el periférico a emplear.

FORMULACIONES DE LOS COMANDOS

CLOAD <nom.>: Carga en memoria el programa almacenado en casete cuyo nombre coincide con el especificado. CSAVE <nom.>: Graba en casete el programa almacenado en memoria, otorgándole el nombre indicado. CLOAD? <nom.>: Comprueba la total coincidencia del programa cargado en memoria con el residente en la unidad de casete. LOAD "<per.>: <nom.>": Carga en memoria el programa indicado que reside en la unidad de almacenamiento externo que se especifica en la zona <per.>. Si se omite la opción <per.>, la unidad implicada será la de disco. SAVE "<per.>: <nom.>": Almacena en el periférico indicado el programa almacenado en memoria, otorgándole el nombre señalado. Cuando la elección de periférico no existe, la grabación se realizará en la unidad de disco. CSAVE "<nom.>, AUTO: Graba en casete el programa indicado, acompañado de un indicativo que hará que el ordenador lo ejecute automáticamente en el momento de cargarlo en la memoria interna. LOAD "<per.|>: <nom.>", R: Formulación del comando LOAD que ordena la autoejecución del programa una vez concluida la carga en memoria.

OBSERVACIONES:

(1) En el CBM 64, la instrucción LOAD como parte de un programa implica la ejecución de dicho programa.

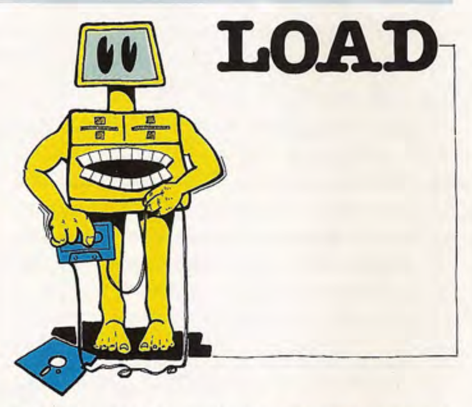
(2) El sistema operativo canaliza la instrucción al periférico adecuado (casete o disco).

LOAD

Carga en la memoria del ordenador un programa residente en una unidad de almacenamiento externo. La opción ", R" ordena su autoejecución automática.

Formato: LOAD "[<*periférico*>*:*] [<*nombre del programa*>]" [*, R*]

Ejemplos: LOAD "1 : EXAMPLE"
LOAD "CAS: "
LOAD "PEPE", R



LOAD, cuya formulación general coincide con:

LOAD "<periférico> : <nombre del programa>", R

Las opciones son las mismas que las del comando SAVE, de tal forma que es necesario especificar el periférico y el nombre del programa. La R que aparece como campo final, separado por una coma, sirve para indicar a la máquina que el programa debe ser ejecutado automáticamente al concluir la operación de carga. Su presencia sustituye virtualmente al comando RUN. Dos ejemplos de formulación correcta son:



Al ejecutar una instrucción LOAD, la información almacenada en el soporte externo es leída por el ordenador y depositada en la memoria central.

```
LOAD "CAS:CARP"
LOAD "2:MIO"
```

Como observación final, cabe añadir que al realizarse la lectura y transferencia del programa desde la unidad de disco a la memoria central, quedará borrado el contenido previo de esta última.

AUTOEJECUCION DE PROGRAMAS

En ciertos casos, es necesario que un programa que se encuentra en una unidad de almacenamiento externo, se autoejecute inmediatamente tras ser cargado en la memoria central. La necesidad puede obedecer a distintas causas. Por ejemplo, puede ocurrir que, por algún motivo, sea necesario encadenar programas (el programa en curso llama a otro que se encuentra en la unidad de almacenamiento). En tal situación, es necesario, o cuando menos conveniente, que el programa llamado se ejecute automáticamente, sin necesidad de que el operador intervenga en el proceso. Esta opción se utiliza con frecuencia para proteger programas profesionales, de tal forma, que la ejecución del referido programa no pueda ser detenida para analizar su codificación y, posiblemente, copiarlo.
Una de las posibilidades para ordenar la autoejecución reside en la opción ", R" que puede acompañar al comando LOAD; por ejemplo:

```
LOAD "CAS:PEPE",R
LOAD "1:PROGRAM",R
```

El tradicional comando de ejecución, RUN, también es utilizable en ciertos dialectos BASIC para este cometido. En tal caso, admite una formulación distinta a la ya conocida. Esta es:

RUN "<periférico> : <nombre>"

Al ejecutarla se consigue el mismo resultado que con LOAD y la opción R. Hay que precisar que la opción <periférico> debe figurar tal y como se indicó en un apartado precedente. Por lo demás, el nombre corresponde al otorgado al programa a ejecutar. Por ejemplo:

```
RUN "CAS:PEPE"
RUN "2:PROG8"
```

LOAD?

Verifica la coincidencia del programa cargado en la memoria interna con el que se encuentra en la unidad de almacenamiento.

Formato: LOAD? "[<*periférico*>*:*] [<*nombre del programa*>]

Ejemplo: LOAD? "CAS: BUT"

Logo (8)

TURTLE GRAPHICS: el color y los movimientos relativos

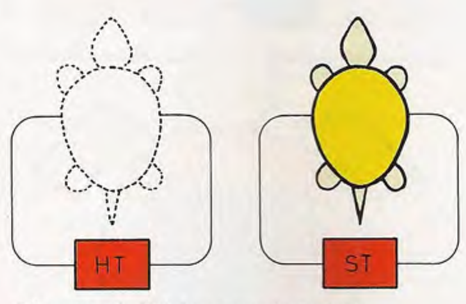
En los capítulos dedicados hasta el momento al trazado de gráficos por medio de la tortuga, se han estudiado algunas de las posibilidades que brinda el LOGO para crear dibujos en la pantalla. A lo largo de esta nueva incursión en el terreno del "turtle graphics", van a ampliarse las herramientas disponibles. Ello permitirá programar dibujos más complejos y espectaculares, a través de un control más flexible y potente de la actividad de la "tortuga gráfica".

LA TORTUGA SE ESCONDE

Una vez terminado un dibujo, puede ser conveniente evitar la presencia de la tortuga. Para conseguirlo, puede desplazarse a la tortuga a una esquina de la pantalla. No obstante, el método más eficaz es ordenarla que se esconda. Para ello, se dispone del comando HT (Hide Turtle). Este comando vuelve invisible a la tortuga, aunque no impide su movimiento ni altera el modo de dibujo seleccionado (PD, PE, PU o PX). Si se desea recuperar su presencia, bastará con teclear la orden opuesta ST (Show Turtle).

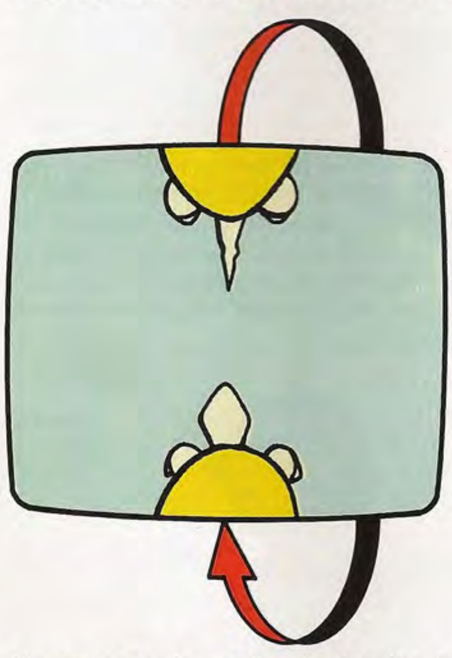
Existe otro método para perder el contacto visual con tan simpático personaje: obligarlo a traspasar los límites de la pantalla. Hay dos alternativas para fijar dichos límites; éstas coinciden con los denominados *modo abierto* y *modo cerrado*.

En el primero, la pantalla muestra únicamente una parte del campo de acción de la tortuga. Si la tortuga sale de esta zona visible o ventana, desaparecerá de la vista del usuario. Este modo se selecciona con el comando WINDOW (ventana).



El comando HT hace que la tortuga desaparezca de la pantalla; a pesar de ello, ésta puede seguir trazando dibujos que serán visibles. Para restaurar su presencia, es preciso utilizar el comando ST.

En el segundo caso, modo cerrado, la tortuga no puede salir de la pantalla. Si intenta evadirse por un extremo, aparecerá de inmediato por el opuesto. Una imagen gráfica de tal situación se obtiene al pensar que la pantalla es una lámina flexible, a la que doblamos y pegamos sus lados opuestos; esto es: el borde izquierdo con el derecho y el superior con el inferior. En tal caso, es obvio que si la tortuga desaparece por el borde superior, reaparecerá de inmediato por el inferior. Y, análogamente, si se ausenta por el lado izquierdo de la pantalla, reaparecerá instantáneamente por el lado derecho. Para optar por este segundo modo se dispone del comando WRAP (enrollar). Es preciso tener cuidado al usar estos dos comandos pues ambos borran la pantalla.



Al operar en modo cerrado, seleccionado con el comando WRAP, la tortuga reaparecerá en la pantalla por el extremo opuesto al que ha desaparecido.

Es posible que, en determinado momento, no se vea a la tortuga y se ignore si está escondida o se encuentra fuera de la pantalla. En éste y en otros casos semejantes, es aconsejable el uso de SHOWNP. El operador SHOWNP devuelve el valor FALSE si se ha ejecutado una orden HT para obligar al quelonio a desaparecer. Su ejecución permite conocer el estado actual, visible o invisible, activado por los comandos ST o HT, respectivamente. Si lo que ocurre es que la tortuga ha salido de la pantalla, pero no se le ha ordenado esconderse, SHOWNP responderá con TRUE.

TIZAS DE COLORES

Por el momento se han confeccionado dibujos, más o menos complejos, controlando los desplazamientos del personaje central del "turtle graphics", pero siempre a base de trazos de un solo color. Desde luego, los dibujos resultarían más atractivos si fuera posible colorearlos.

En primera instancia, las posibilidades de elección de color dependen del ordenador

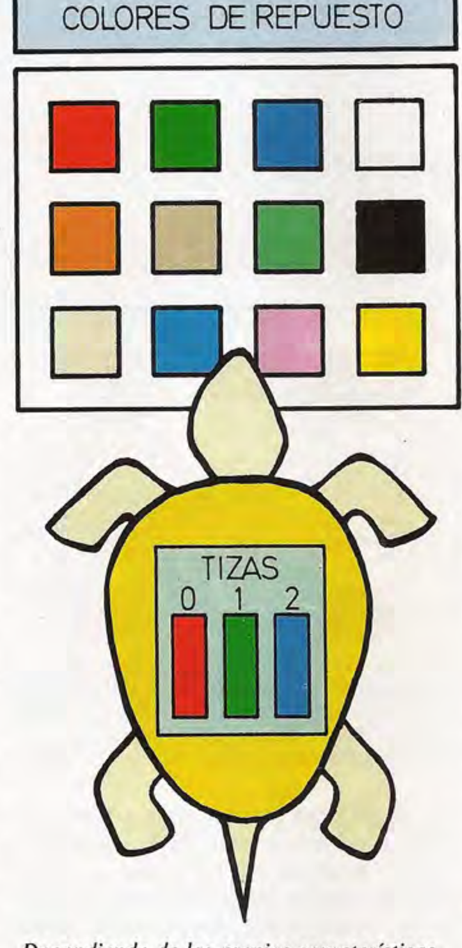
que se utilice. Al respecto, es preciso consultar el manual propio del aparato con objeto de aprovechar las distintas opciones que brinda. Una vez precisadas las posibilidades de color del equipo, hay que poner en práctica los medios que incorpora el LOGO para controlar las tizas de color.

En principio, hay que señalar que la tortuga puede transportar más de una tiza. Estas están numeradas para facilitar su identificación.

Para cambiar de tiza se utiliza la orden SETPN, seguida por el número de tiza deseado. Dicho comando permite trazar dibujos que aparecerán en la pantalla con el color seleccionado. La nueva tiza seguirá utilizándose hasta que se cambie de nuevo su color al ejecutar un nuevo comando SETPN. El número habitual de tizas que puede transportar la tortuga se eleva a tres. Por otra parte, en cualquier momento se puede identificar el número de tiza en uso, por medio del operador PN; su ejecución indica cuál es el número de la tiza en acción.

Lo más frecuente es que el ordenador permita utilizar un número de colores superior a tres. Para hacer uno de los restantes colores será necesario cambiar el juego de las tres tizas que puede viajar con la tortuga. El comando que reasigna los colores de las tizas es SETPC. SETPC exige dos entradas: la primera debe coincidir con el número de tizas a reasignar, mientras que la segunda debe ser un número que identifique a uno de los colores que permite utilizar el ordenador.

El operador PC constituye una herramienta de ayuda para el usuario en este punto, puesto que al utilizarlo acompañado por un número de tiza, la máquina responderá con el número del color asignado a la misma.



Dependiendo de las propias características del ordenador, la tortuga será capaz de utilizar un mayor o menor número de tizas de colores. En todo caso, hay que tener en cuenta que ésta sólo puede llevar un número reducido de tizas. Para que le sea posible utilizar los restantes colores, debe intercambiar las correspondientes tizas con las que lleva "puestas" en ese preciso instante.

CAMBIO DEL FONDO

Al emplear tizas de colores puede ocurrir, en algún caso, que se esté pintando con una tiza cuyo color coincide con el color del fondo. De darse esta situación, los dibujos no resultarán visibles. Para solventar esta situación, el LOGO permite alterar el color del fondo de la pantalla.

El comando adecuado es SETBG (SET BackGround). Al igual que ocurre con las tizas, los colores disponibles dependerán de la gama que ofrezca el ordenador. SETBG debe ir seguido por el correspondiente número de color. Esta posibilidad permite alterar con comodidad el escenario del dibujo: verde para un paisaje campestre, azul para un fondo marino o negro para una escena galáctica.

De nuevo, el usuario puede averiguar en cualquier momento cuál es el color del fondo. Para ello cuenta con el operador BG (BackGround). BG devuelve el número correspondiente al color actual de la pantalla.

El empleo conjunto del comando SETBG y del operador BG permite múltiples posibilidades. Por ejemplo:

```
SETBG SUM 1 BG
```

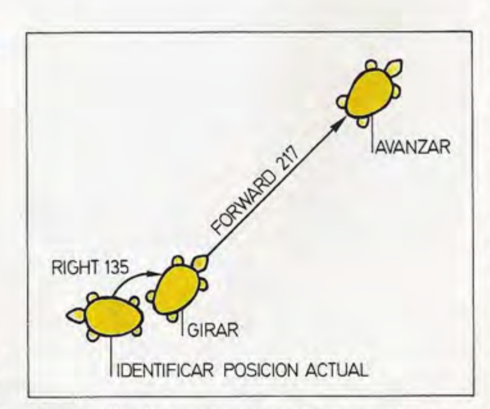
recupera el número de color, le suma una unidad, y utiliza el resultado para definir el nuevo color de fondo. Con ello, cada vez que se ejecuta esta orden se altera el color de la pantalla, pasando al color cuyo código coincide con el siguiente en número.

DE LO RELATIVO A LO ABSOLUTO

Los desplazamientos producidos por las órdenes FORWARD y BACK se denominan relativos. Este apelativo se refiere al hecho de que la posición final o de destino de la tortuga depende de su situación de partida. Asimismo, la orientación de la tortuga por efecto de LEFT o RIGHT también es relativa: una misma orden LEFT o RIGHT dejará a la tortuga en una orientación distinta según sea la orientación inicial o anterior a recibir la orden. Si lo que se desea es situar a la tortuga en un punto determinado y con una orientación específica, es preciso seguir una serie de pasos:

—Identificar la posición y orientación actuales.

—Orientar a la tortuga en dirección al nuevo punto.



Secuencia de pasos necesarios para situar a la tortuga en un determinado punto de la pantalla.

—Avanzar el número necesario de unidades.
—Girar hasta la orientación deseada.

El proceso puede simplificarse mediante el uso previo del comando HOME (traslado al origen), con lo que la posición y orientación inicial de la tortuga resultarán conocidas (las iniciales al encender la máquina). En todo caso, este método supone el paso de la tortuga por el centro de la pantalla.

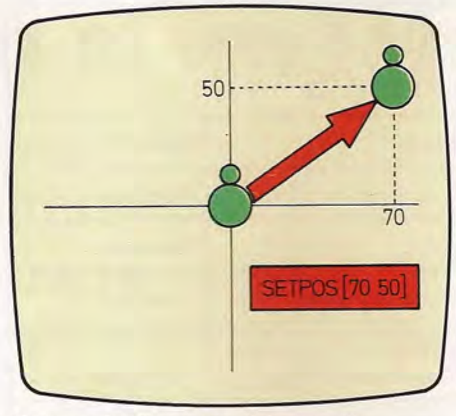
Las órdenes para el desplazamiento absoluto eliminan toda esta suerte de inconvenientes. Con un solo comando, SETPOS, es posible situar a la tortuga en la posición de la pantalla que se desee. Esta debe especificarse por medio de sus coordenadas, expresadas en forma de lista. Dichas coordenadas serán las cartesianas (eje X y eje Y, en ese mismo orden) referidas, por lo general, al centro de la pantalla.

Por ejemplo, la orden SETPOS [50 25] coloca a la tortuga a 50 unidades hacia la derecha y 25 hacia arriba del origen o centro de la pantalla, sea cual fuere su posición actual. Como es habitual, si la tortuga tiene la tiza activada (bajada), aparecerá en la pantalla el trayecto recorrido, que coincidirá con una línea recta. En caso de duda, es posible conocer las coordenadas del punto en el que está situada la tortuga. Al efecto, se dispone del operador POS; éste devuelve una lista conteniendo dichas coordenadas. El uso de POS permite almacenar las coordenadas de un punto de la pantalla para, posteriormente, utilizarlas en orden a ubicar en él a la tortuga por medio de SETPOS. Por ejemplo:

```
PENUP
LEFT 123
FORWARD 73
MAKE "PUNTO POS
HOME
BACK 29
SETPOS :PUNTO
```

En el ejemplo, se selecciona una posición cualquiera por medio de LEFT y FORWARD. Las coordenadas de esta posición se almacenan en la variable PUNTO. A continuación, se utilizan las órdenes HOME y BACK para alejarse del referido punto. Por último, SETPOS :PUNTO restaura a la tortuga en el lugar elegido.

Las dos órdenes mencionadas ejecutan un desplazamiento absoluto definido por las coordenadas. El uso de coordenadas permite definir un nuevo tipo de movimiento: el desplazamiento intermedio en-



Efecto sobre la tortuga del comando SETPOS: desplaza a la tortuga en modo absoluto a la posición cuyas coordenadas se incluyen en la lista que acompaña al comando.

tre el absoluto y el relativo. Un desplazamiento de este tipo se obtiene variando únicamente una de las coordenadas. El siguiente procedimiento ilustra tal posibilidad; traslada la tortuga al punto cuya abscisa (posición X) se indica, sin variar su ordenada (coordenada Y):

```
TO SETPOSX :X
MAKE "Y LAST POS
SETPOS [ :X :Y]
END
```

El cometido del operador LAST es extraer la ordenada de la posición actual; ésta se utiliza al definir la nueva posición, asignando a dicho valor a :Y. Un procedimiento complementario a éste será el que altere la coordenada Y, manteniendo la posición horizontal (coordenada X). Respecto al ejemplo precedente, el nuevo procedimiento sustituirá las X por Y, las Y por X y LAST por FIRST.

A continuación se introducen nuevas órdenes que permitirán simplificar al máximo este procedimiento. Su cometido es facilitar la reutilización de alguna de las coordenadas de la posición inicial de la tortuga.

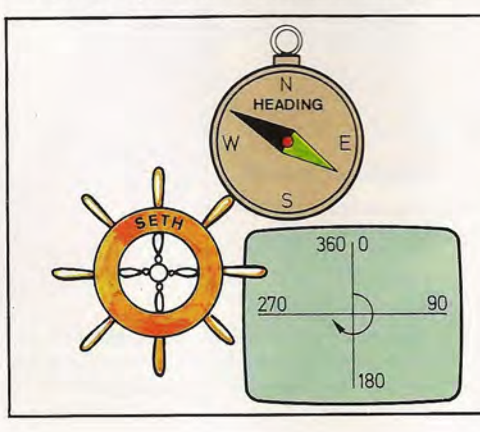
El lenguaje LOGO incorpora dos operadores útiles al efecto, estos son XCOR e YCOR. El primero devuelve el valor de la abscisa (coordenada X), e YCOR hace lo propio con el valor de la ordenada (coordenada Y). Utilizando esta nueva opción el procedimiento anterior quedará de la siguiente forma:

```
TO SETPOSX :X
MAKE "Y YCOR
SETPOS [ :X :Y]
END
```

O mejor aún:

```
TO SETPOSX :X
SETPOS [ :X :YCOR]
END
```

Los dos próximos comandos simplifican aún más el procedimiento. En realidad lo reducen a la nada, puesto que realizan específicamente dicho cometido. Estos son SETX y SETY. Con SETX se varía la



El comando SETH permite orientar a la tortuga en la dirección y sentido definido por el ángulo que constituye su dato de entrada. HEADING es un operador capaz de indicar la orientación actual de la tortuga, expresada en grados.

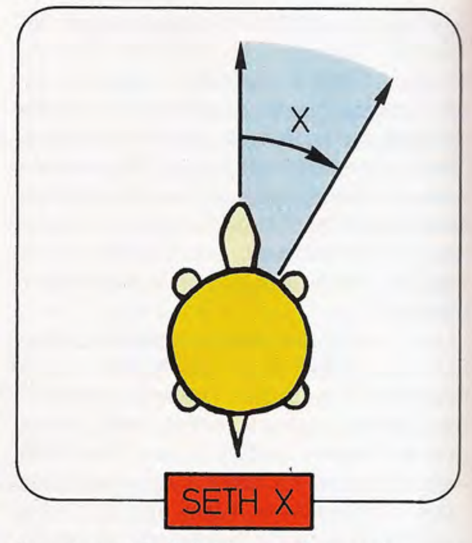
TABLA DE ORDENES DEL "TURTLE GRAPHICS" (1)

Instrucción	Cometido	Operador/comando
HT	Esconde a la tortuga	Comando
ST	Muestra a la tortuga	Comando
SHOWNP	Devuelve TRUE si la tortuga es visible	Operador
WINDOW	Activa el modo en el que la pantalla sólo representa una parte del campo de la tortuga	Comando
WRAP	Activa el modo cerrado en el que la tortuga no puede salir de la pantalla	Comando
SETPN<NT>	Utiliza la tiza número <NT>	Comando
SETPC<NT><NC>	Adjudica a la tiza <NT> el color <NC>	Comando
SETBG<NC>	Pone el fondo de color <NC>	Comando
PN	Devuelve el número de tiza utilizado	Operador
PC<NT>	Devuelve el número del color de la tiza <NT>	Operador
BG	Devuelve el número del color de fondo	Operador

<NT>: número de tiza.
<NC>: número de color.

TABLA DE ORDENES DEL "TURTLE GRAPHICS" (2)

Instrucción	Cometido	Operador/comando
SETPOS<lista>	Sitúa a la tortuga en la posición dada por las coordenadas de la lista	Comando
POS	Devuelve una lista conteniendo las coordenadas del punto donde se encuentra la tortuga	Operador
XCOR	Devuelve el valor de la coordenada X del punto actual	Operador
YCOR	Devuelve el valor de la coordenada Y del punto actual	Operador
SETX<número>	Sitúa a la tortuga en el punto de coordenada X, dada por <número>, sin variar su coordenada Y	Comando
SETY<número>	Sitúa a la tortuga en el punto de coordenada Y, dada por <número>, sin variar su coordenada X	Comando
SETH<número>	Sitúa a la tortuga con la orientación dada por <número>	Comando
HEADING	Devuelve la orientación actual de la tortuga	Operador



La orientación definida por el comando SETH afecta a la tortuga en términos absolutos; esto es: la tortuga girará hasta orientarse en el ángulo indicado, medido a partir de la vertical y en el sentido de las agujas del reloj.

posición horizontal de la tortuga sin alterar la vertical, mientras que SETY mantiene la posición horizontal (coordenada X) modificando la vertical (coordenada Y).

CAMBIO DE ORIENTACION

Las órdenes de movimiento presentadas en los párrafos precedentes, conducen a un cambio de posición que mantiene la *orientación* inicial de la tortuga. Los comandos LEFT y RIGHT, anteriormente comentados, se utilizan para un cambio relativo de orientación: ordenan el giro de la tortuga en un determinado ángulo a partir de la orientación actual.

Para ordenar el posicionamiento en modo absoluto se dispone de SETH (SET Heading). El ángulo de orientación debe ser expresado en grados, tomando como origen (generalmente) el eje vertical, en sentido hacia arriba, y realizándose el giro en el sentido de las agujas del reloj. Así, por ejemplo, SETH 90 coloca a la tortuga mirando hacia la derecha.

Existe, asimismo, un operador que devuelve el ángulo de orientación actual de la tortuga. Este es HEADING; un operador que responde expresando la orientación en los términos antes señalados.

Introducción al MP/M

Características generales del sistema operativo multiusuario

Al igual que el CP/M, el MP/M es un sistema operativo para microordenadores basados en los microprocesadores 8080 y 8085 de Intel y Z80 de Zilog. Su principal característica es que gestiona el trabajo de varios usuarios con la misma unidad central de proceso (CPU). En definitiva, introduce el concepto de multiusuario, justificando plenamente las siglas que le dan nombre: "Multi-Programming Monitor".

Por lo tanto, la configuración física necesaria requiere la presencia de un solo microprocesador, de los anteriormente mencionados. Con la colaboración del MP/M, éste será capaz de gestinar la información procedente de los diversos usuarios, y estará en situación de controlar a los distintos dispositivos periféricos que formen parte del sistema: terminales, impresoras, unidades de disco... Para llevar a cabo estas funciones, el sistema operativo MP/M exige una capacidad de almacenamiento en la memoria central del ordenador de al menos 32 Kbytes de RAM. Además, requiere la presencia en el equipo de un reloj de tiempo real, adecuado para contabilizar el tiempo que cada proceso desencadenado por un usuario puede estar accediendo a la CPU.

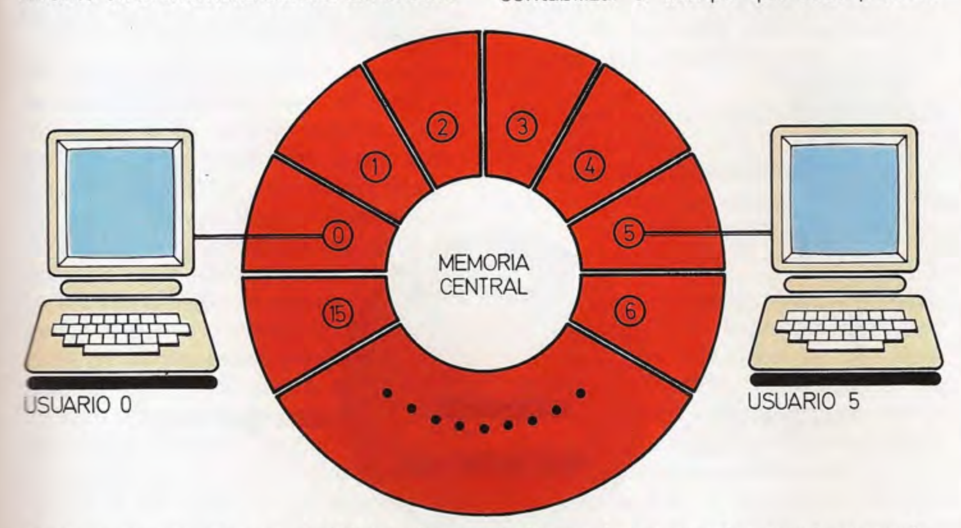
El MP/M es un sistema operativo multiusuario, capaz de soportar el trabajo de hasta 16 usuarios (numerados del 0 al 15) conectados al mismo ordenador.

En términos generales, el usuario se encontrará virtualmente inmerso en un sistema monousuario de tipo CP/M, ya que el sistema operativo MP/M utiliza el mismo módulo BDOS que forma parte del CP/M. Tal circunstancia permite una total compatibilidad con los programas desarrollados bajo el sistema operativo CP/M; éstos pueden ser ejecutados en un entorno multiusuario sin necesidad de someterlos a ningún cambio.

LOS USUARIOS DEL SISTEMA

El número de usuarios que puede soportar el sistema operativo MP/M se eleva a un máximo de 16. Cada uno de los usuarios está referenciado por un número de identificación propio (del 0 al 15). Al respecto, hay que señalar que el número de usuario no debe confundirse con el número que tienen asignado los diversos terminales o puestos de trabajo del sistema. Aunque ambos números pueden coincidir en algún caso, se refieren a conceptos distintos; un determinado usuario no tiene por qué estar asignado permanentemente a una misma pantalla física, ni viceversa.

Cada usuario dispone de una zona de almacenamiento en disco reservada para su uso exclusivo. Este es "dueño" de los ficheros y programas que residan en la misma. Los distintos usuarios son totalmente independientes, hasta el extremo que dos de ellos pueden tener almacenados en sus respectivas zonas dos programas idénticos, e incluso con el mismo nombre. Por supuesto, en cada situación



En los equipos regidos por el MP/M, cada usuario dispone de una zona de memoria reservada para su uso exclusivo. La única excepción aparece en la zona reservada al usuario 0; ésta última, utilizada por el sistema, no es exclusiva, sino que todos los usuarios pueden acceder a la misma en modo lectura.

se ejecutará el programa que corresponda, según sea un usuario u otro quien llame al programa en cuestión.

La zona reservada para el usuario cero es utilizada por el sistema. Esta no es exclusiva, sino que es de uso común. Todos los usuarios (con los números del 1 al 15) tienen la posibilidad de acceder a ella en modo de lectura, esto es: pueden examinar y ejecutar los programas que se encuentran en ella, aunque no pueden en ningún caso alterar su contenido.

Los diversos programas almacenados en el usuario cero tienen, por lo general, un carácter complementario de los comandos y utilidades del sistema. Suelen ser de uso frecuente por parte de los diferentes usuarios. Este hecho contribuye a optimizar el espacio ocupado en disco, ya que se evita la posible duplicidad de programas al estar centralizadas la mayoría de las utilidades.

Cada usuario tiene asignado un terminal; desde éste puede acceder a los recursos del sistema al ordenar que se ejecute cierto comando o programa.

Cuando se activan varios programas a la vez (operación multitarea), el sistema operativo se encargará de organizar la compartición del tiempo de acceso a la CPU entre los diversos procesos demandantes, de acuerdo a su prioridad; desde luego, los distintos procesos compartirán de igual forma los dispositivos periféricos asociados al sistema.

EL CONCEPTO DE PROCESO

El concepto de programa como un conjunto de códigos ejecutables, aislados y autosuficientes, queda obsoleto en el ámbito del MP/M. En general, los programas necesitan interaccionar con el sistema operativo para establecer comunicación y operar con los distintos periféricos. Así pues, un proceso no es simplemente un bloque estático de código ejecutable, sino que además de la propia ejecución del código del programa debe gobernar la ejecución del código del sistema operativo requerido por el programa, pasando, así, a trabajar de una forma esencialmente dinámica.

Los procesos se generan al ejecutar una carga mediante el comando LOAD para crear un fichero objeto; automáticamente, se asocia un proceso al nuevo programa ejecutable que acaba de ser creado. Por consiguiente, es el proceso en vez del programa el que controla los accesos a los diversos recursos del sistema.

En el marco del MP/M se consideran tres categorías de programas:

—Programas CP/M.

—Procesos propios del sistema.

—Procesos residentes del sistema o "Resident System Processes" (RSP).

La primera categoría engloba a programas CP/M clásicos, que una vez cargados e inicializados son asociados a un proceso sobre el que actuará el MP/M. La segunda incluye a aquellos procesos que realizan las tareas del sistema operativo; por ejemplo, al *Intérprete de Comandos de Línea* CLI (del inglés "Command Line Interpreter"), cuya misión es cargar e inicializar los programas de los usuarios. La última categoría está integrada por procesos que pueden ser integrados opcionalmente en la generación del sistema. Este tercer grupo permite la inclusión de procesos escritos por el usuario; procesos que pueden paliar alguna carencia o adaptar más perfectamente el sistema operativo originalmente suministrado.

Atendiendo a la naturaleza de las funciones que tienen asignados, el MP/M se divide en tres módulos: XIOS, XDOS y BDOS.

DESCRIPCION FUNCIONAL DEL MP/M

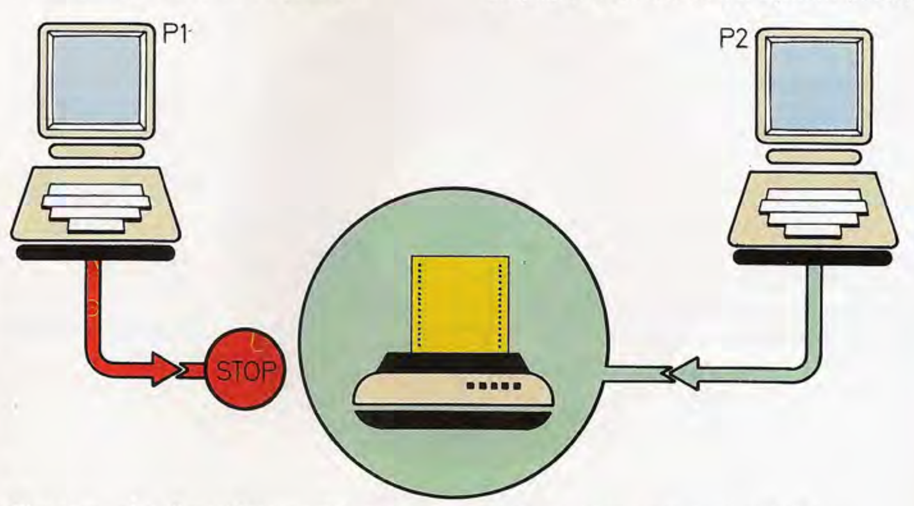
Uno de los principales objetivos en el diseño del sistema operativo MP/M, fue lograr la compatibilidad con el software ya desarrollado para CP/M. De ahí que la estructura de los diversos módulos que componen el sistema operativo multiusuario sea muy parecida a la propia de los módulos del CP/M. Así pues, más que diferencias reales, los módulos del MP/M coinciden con ampliaciones adecuadas para contemplar las nuevas funciones introducidas.

Funcionalmente, el MP/M está compuesto por tres módulos:

—Sistema operativo de disco básico o *Basic Disk Operating System (BDOS).*

—Sistema operativo de disco ampliado o *Extended Disk Operating System (XDOS).*

—Sistema operativo de entrada-salida ampliado o *Extended I/O System* (XIOS).



El acceso a los dispositivos periféricos comunes al sistema se canaliza por medio de interrupciones. Estas dan prioridad al usuario que primero solicite el recurso, excluyendo momentáneamente a los restantes procesos que invoquen al periférico en cuestión.

EL MODULO BDOS

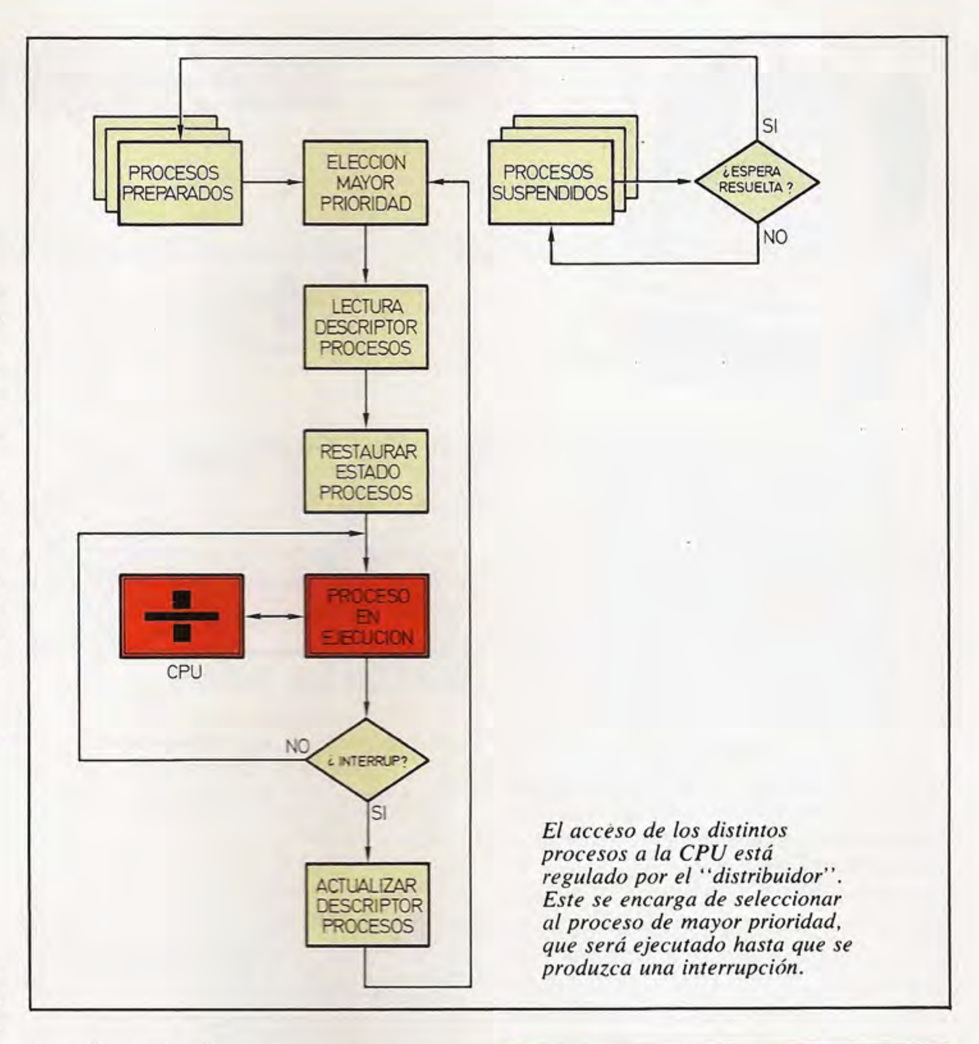
Este módulo contiene las funciones lógicas para la gestión de ficheros en disco y para el gobierno de las distintas consolas o terminales. Es compatible con la versión monousuario. Ello significa que, en la mayor parte de los casos, pueden ejecutarse programas que hacen llamadas al módulo BDOS del CP/M sin tener que modificarlos. No obstante, el módulo BDOS está ampliado respecto a la versión monousuario; una ampliación con dos objetivos básicos: hacer frente a los accesos múltiples que se producen en los archivos cuando se actúa en un entorno multiusuario, y llegar a soportar el uso de múltiples pantallas y dispositivos de presentación.

EL MODULO XDOS

Su misión es proporcionar al sistema operativo MP/M las herramientas adecuadas para gestionar la posibilidad de multiprogramación. Dentro de este módulo caben tres zonas funcionales: el *núcleo lógico del sistema* que se encarga de supervisar la ejecución de los diferentes procesos que compiten por la CPU y los recursos del sistema; las diversas ampliaciones referidas a las funciones de getión de ficheros y, por último, las utilidades necesarias para leer los comandos del usuario y ejecutar los programas que correspondan.
Dentro del *núcleo lógico del sistema* se distinguen varios submódulos, entre los que destaca el *distribuidor de procesos*.
La transferencia de los recursos de la CPU de un proceso a otro corre a cargo de una parte del núcleo denominada *distribuidor*. Esta se encarga de asignar el tiempo de disfrute de la CPU a cada proceso, en base a una estructura de datos, asociada a cada proceso, llamada *descriptor de procesos*. El descriptor de un proceso contiene las características propias del mismo y es utilizado por el distribuidor para apuntar cómo quedó un proceso al agotarse su tiempo de acceso a la CPU, o para examinar el estado de un proceso que va a empezar a consumir tiempo de CPU.
Los posibles estados de un proceso son tres:

—Preparado o esperando para utilizar la CPU.
—En ejecución o consumiendo recursos de la CPU.
—Suspendido o aguardando para acceder a algún recurso del sistema distinto de la CPU, o por efecto de algún acontecimiento particular.

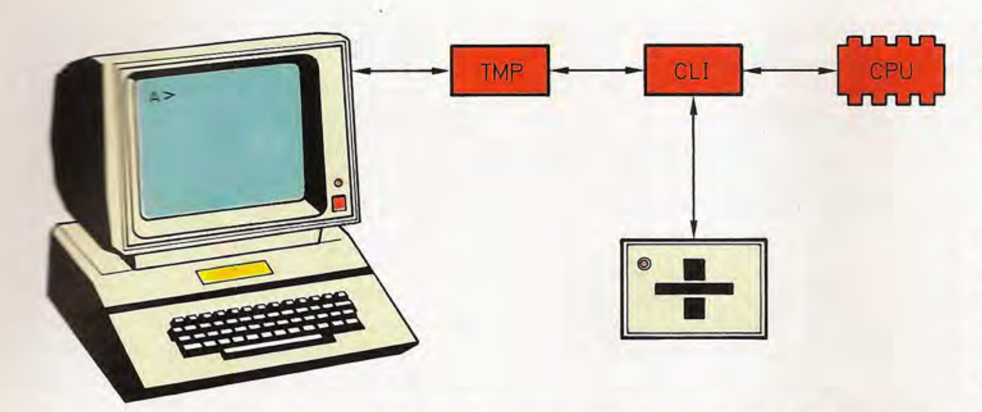
Así pues, el distribuidor va permitiendo el acceso a la CPU de acuerdo a la prioridad indicada en el descriptor de procesos, hasta que se produzca la interrupción y le llegue el turno al siguiente proceso de superior prioridad. Este método de trabajo implica que siempre debe existir un proceso en ejecución; al respecto, el sistema mantiene "en observación" a un proceso (IDLE process), con una prioridad muy baja, que se ejecutará siempre que no haya otro proceso activo.



El acceso de los distintos procesos a la CPU está regulado por el "distribuidor". Este se encarga de seleccionar al proceso de mayor prioridad, que será ejecutado hasta que se produzca una interrupción.

GESTION DE COLAS

La comunicación entre los distintos procesos, el sincronismo de su ejecución y la exclusión mutua entre procesos, son tareas que se pueden llevar a cabo poniendo en práctica una estructura especial de datos llamada *cola*. Esta estructura es similar a un fichero de disco, aunque con la particularidad de que siempre reside en la memoria central. Sobre ella se puede actuar de forma análoga a un fichero convencional: se puede abrir, cerrar, leer, escribir y borrar.
En las *colas*, los procesos pueden leer o escribir mensajes de dos formas: condicional o incondicional. Si un proceso efectúa una lectura incondicional de una cola vacía, el proceso en cuestión queda suspendido, pasando a una cola especial (Dequene List), hasta que otro proceso es-



El nexo entre el usuario y los comandos residentes en la unidad central, se establece a través del "procesador de mensajes del terminal" (TMP) y del "intérprete de comandos de línea" (CLI).

criba en dicha cola. Análogamente, si un proceso trata de escribir en una cola llena, éste se verá suspendido, pasando a otra cola especial (Enquene List), hasta que quede sitio libre para realizar la escritura.
El MP/M utiliza estas dos colas especiales, en la forma indicada, para lograr la sincronización de los procesos. Si la lectura o escritura se realiza de forma condicional, los procesos no pasarán a las referidas colas especiales, sino que el sistema generará un mensaje de error cuando no es posible efectuar la operación. La exclusión mutua de dos procesos se gestiona mediante colas cuyo nombre empieza por la letra MX y en cuyo interior los mensajes tienen una longitud nula; de esta forma se garantiza el acceso de un solo proceso a un recurso determinado para evitar posibles conflictos.

GESTION DE BANDERAS

El núcleo lógico del MP/M utiliza banderas o indicadores lógicos para señalar o sincronizar procesos según haya ocurrido un proceso o no. Estas banderas permiten al sistema generar interrupciones físicas que se producen desde el hardware e interaccionan con la parte física de la máquina. Hay que tener en cuenta que tales indicadores están especialmente diseñadas para interaccionar con el hardware, de ahí que no sea conveniente utilizarlos en aplicaciones software, salvo que se tenga un gran conocimiento de su implicación o sea estrictamente necesario; en general, la comunicación y el sincronismo entre procesos se resuelve mejor mediante la técnica de colas.

GESTION DEL TIEMPO

La exacta contabilización del tiempo por parte del sistema operativo MP/M, necesaria para su correcto funcionamiento, está en manos de dos procesos del sistema: TICK y CLOCK.
El primero de ellos, TICK, reactivado cada 20 milisegundos, determina el tiempo de acceso a la CPU permitido para cada proceso activo. La frecuencia de reactivación no es aconsejable que sea muy elevada, puesto que se perdería mucho tiempo guardando y restaurando procesos que abandonan y retoman el acceso a la CPU. No obstante, tampoco es conveniente que sea muy baja, ya que un proceso podría adueñarse de la CPU durante un largo período y paralizar a los restantes.
El segundo proceso del sistema, especializado en este cometido, es CLOCK. Este es activado cada segundo y su función es la de mantener e ir incrementando la fecha; de esta forma el sistema tiene un conocimiento inmediato y constante del año, mes, día, hora, minuto y segundo en curso.
Con estas funciones, el sistema es capaz de ofrecer la fecha, retardar la ejecución de procesos un determinado período de tiempo y regular un programa para que se cargue desde el disco y se ejecute en un preciso instante.

LA COMUNICACION CON LAS CONSOLAS

En el MP/M, cada consola tiene asociado un proceso de mensajes del terminal, TMP (del inglés "Terminal Message Process"), numerados del 0 al 15 y asignados a cada terminal en el instante de generar el sistema; su presencia permite soportar las líneas de comandos aceptadas por cada terminal.
Este proceso, al igual que cualquier otro, tiene su *descriptor de proceso,* localizado en el segmento TMPD. DAT, y su código único para cada TMP (código reentrante) reside en el módulo ejecutable TMP.SPR.
El propio sistema operativo se encarga de mantener los buffers, pilas y variables locales necesarias para cada TMP.
El TMP puede leer comandos tanto del terminal como de unos ficheros especiales de tipo SUB, residentes en disco, y que contienen las distintas líneas de comandos. Una vez leído el comando, el TMP asigna la consola a otro proceso llamado *intérprete de líneas de comandos,* CLI ("Command Line Interpreter"). En ese instante, el CLI toma el control del terminal, interpreta el comando y activa el programa transitorio o programa residente del sistema (RSP) requerido. Tras la ejecución del referido programa, el CLI devuelve el control de la pantalla al TMP.

EL MODULO XIOS

El módulo XIOS no es más que una extensión del BIOS del CP/M, cuyo objetivo es hacer frente a las capacidades de multiterminal y multiprogramación del MP/M. Como es lógico, varía de uno a otro modelo de ordenador; hay que tener en cuenta que este módulo incluye programas específicos que sirven de mero nexo entre la parte lógica y el entorno material (hardware), y éste suele cambiar de un fabricante a otro.
Las principales funciones que realiza el XIOS son: manejo de los puertos o accesos para comunicación serie y paralelo, tratamiento de las interrupciones, acceso a disco y gestión de tiempos.

Hojas electrónicas

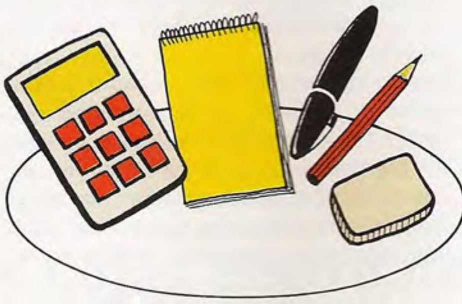
La solución informática a los problemas de "lápiz, papel y calculadora"

Existen muchas aplicaciones que tradicionalmente han estado apartadas de la trayectoria hacia la mecanización. Por ejemplo, a nivel personal, el control de pagos a colegios o la contabilización de ingresos laborales, y a nivel empresarial, la planificación financiera o la optimización de resultados.

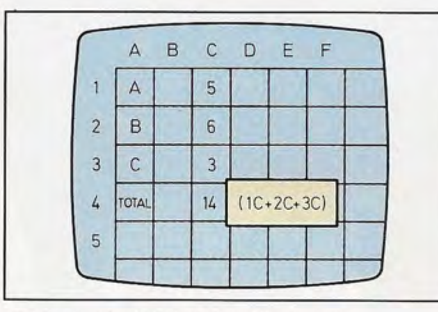
Cuando alguien pensaba en confeccionar un programa para ordenador destinado a resolver alguna de las aplicaciones señaladas, acababa, generalmente, desechando la idea, ya que los métodos empleados eran (o podían ser) distintos día a día. Los datos manejados eran personales y a veces confidenciales, y el volumen de información no era demasido grande. Si embargo, al sumar los datos manejados en varias ocasiones, su volumen podía llegar a resultar considerable. En definitiva, a este tipo de problemas se les denomina *"de lápiz, papel y calculadora"*.

ORIGEN DE LAS HOJAS ELECTRONICAS

En la Universidad de Harward se consideró la posibilidad de desarrollar una aplicación informática capaz de resolver cualquiera de los problemas de lápiz, papel y calculadora. Evidentemente, para que el programa fuera bien aceptado debía ser muy versátil, de forma que no se limitara a un campo de actuación exclusivo. Sólo sus futuros usuarios debían ser los encargados de buscar aplicaciones concretas al programa. Otra cualidad exigible era su sencillez de uso. El usuario medio del producto no sería un especialista en Informática, y su nivel de conocimientos podía ser muy diverso.

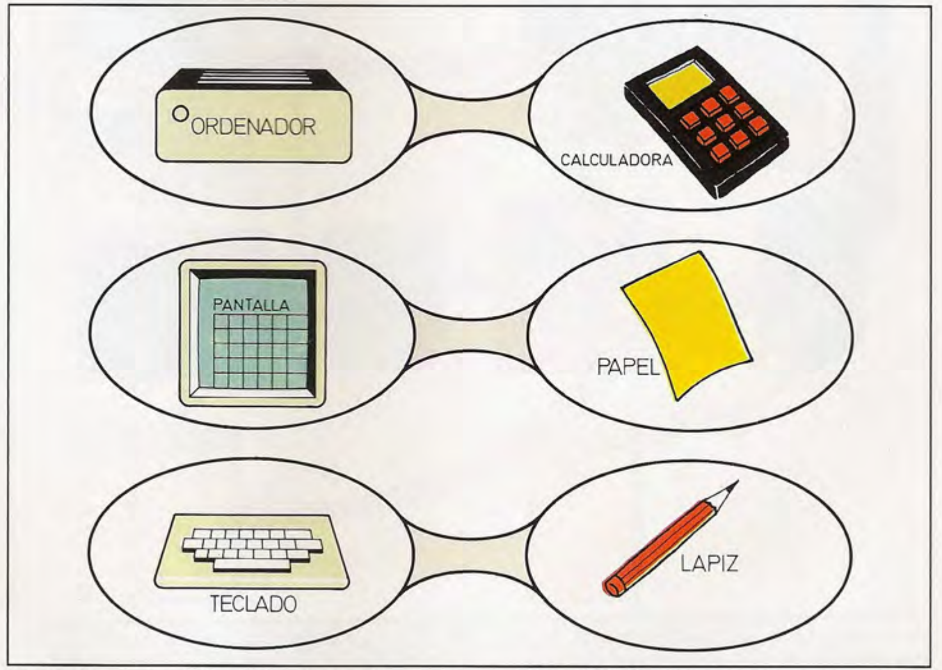


Tradicionalmente, muchos problemas han estado reservados para su resolución manual, a base de lápiz, papel y calculadora. Con la creación de las hojas electrónicas, son muy numerosas las aplicaciones de esta categoría que han pasado a manos del ordenador personal.



La hoja electrónica constituye una herramienta idónea para solucionar problemas numéricos, resueltos hasta ahora a base de tabular columnas de datos en una retícula de papel, establecer relaciones matemáticas entre los datos y evaluar los resultados con una calculadora.

Bob Frankston y Dan Brickon, en San José de California, llegaron a desarrollar una aplicación que denominaron VISICALC. Esta reunía todas las exigencias anteriores. A nivel comercial, la idea resultó tan brillante que a los pocos meses de su



En la alternativa informática, la función que desempeñaba el lápiz pasa a quedar en manos del teclado; a su vez, el papel se ve sustituido por la pantalla y la calculadora por la perfecta coordinación de la aplicación de hoja electrónica y el ordenador.



lanzamiento formaron una empresa con el único objetivo de mantener y distribuir su aplicación. Posteriormente, surgieron muchos productos similares, hasta llegar a la gran variedad de aplicaciones semejantes que se encuentran en el mercado actual.
Como ya se ha apuntado, una de las características más importantes de los problemas que se pretenden resolver con este tipo de aplicaciones, es que los usuarios no quieren verse "atrapados" por un modelo rígido. De hecho, cabía suponer que no habían optado mecanizaciones particularizadas por no perder la libertad de que disponían trabajando sobre una hoja de papel.
Intentando resaltar esta cualidad y buscando un nombre comercial pegadizo y competitivo, este software de aplicación fue bautizado como: Hojas Electrónicas.
Los resultados de aplicar este producto a ordenadores personales han sido tan óptimos que, en la actualidad, muchas de las aplicaciones de planificación, que tradicionalmente se explotaban en grandes ordenadores, han pasado a resolverse cómodamente con un microordenador y la correspondiente hoja electrónica. Con la ventaja adicional de que los directivos de la empresa son usuarios finales del producto, no dependiendo, por lo tanto, más que de su propia capacidad de gestión para tomar las decisiones sobre el cauce que debe seguir su empresa.



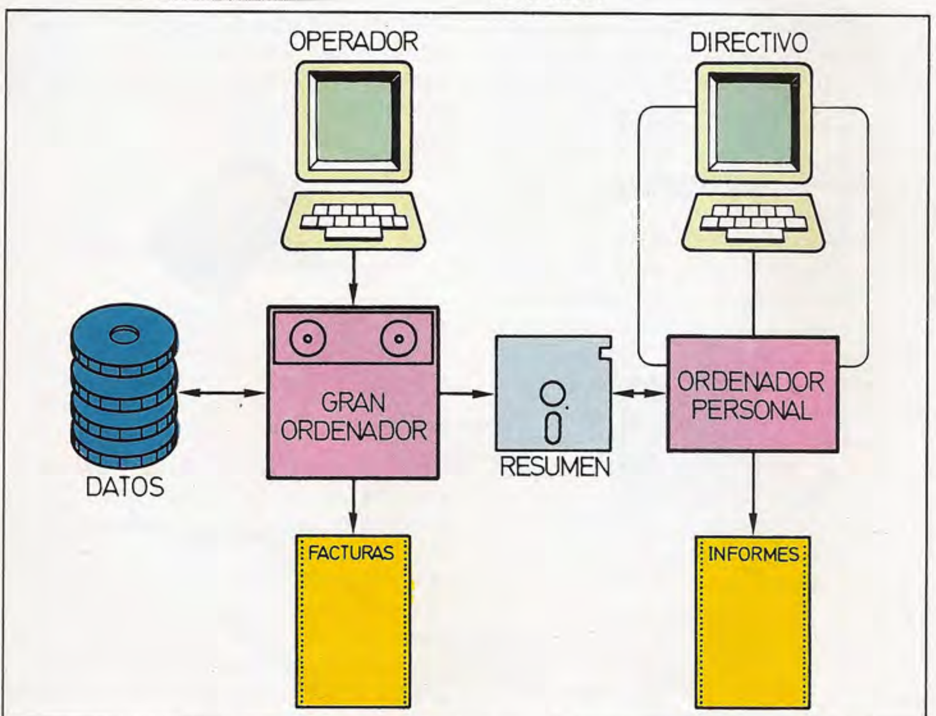
Planificación financiera, evaluación de modelos económicos, análisis de costes, preparación de ofertas..., todas ellas son actividades cuya resolución puede acometerse con el apoyo de una hoja electrónica.

¿EN QUE CONSISTE UNA HOJA ELECTRONICA?

Desde el punto de vista físico, una hoja electrónica es un programa o paquete de programas dedicado a resolver problemas del siguiente tipo:

- Planificaciones financieras de mercado, de personal, etc.
- Gestión de procesos de fabricación.
- Seguimiento de costos.
- Preparación de ofertas.
- Estadísticas diversas.
- Y otras muy diversas.

Normalmente, estos programas se ejecutan en ordenadores personales, aunque también es posible explotarlos en miniordenadores e incluso en grandes ordenadores. La tendencia actual en empresas de tipo medio o grande, consiste en manejar los datos operativos de la empresa mediante aplicaciones tradicionales de gestión, en ordenadores grandes; y una vez procesados todos los datos, transmitir un resumen significativo de los datos manejados a un ordenador personal, para que en ese momento la dirección de la empresa pueda manejar cómoda y rápidamente esa información resumida, ya sea mediante hojas electrónicas, procesadores de texto...
Un ejemplo típico de este tipo de procesos puede ser el siguiente: supóngase una aplicación tradicional de facturación de servicios a clientes, que maneje un gran volumen de información. El producto final de esta aplicación serán las facturas propiamente dichas y un resumen de facturación que puede incluir, entre otras cosas, facturación por cliente, facturación por producto, facturación por empleado, etc. Estos resúmenes no deben ser estáticos, ya que en determinado momento



Un ejemplo típico de transferencia de información, para su posterior tratamiento por medio de una hoja electrónica. El gran ordenador envía un resumen de la facturación de la empresa a un ordenador personal; adoptando una hoja electrónica como herramienta de trabajo, el directivo estará en condiciones de utilizar el ordenador personal para elaborar análisis económicos y emitir informes al respecto.

	A	B	C	D	
1	EMPLEAD.	HORAS	COSTE	IMPORTE	
2	SAEZ	15	1000	15000	
3	PEREZ	10	1200	12000	
4	RUIZ	20	800	16000	
5	TOTAL			43.000	

	A	B	C	D	
1	LITERAL	LITERAL	LITERAL	LITERAL	
2	LITERAL	DATO	DATO	FORMULA	
3	LITERAL	DATO	DATO	FORMULA	
4	LITERAL	DATO	DATO	FORMULA	
5	LITERAL			FORMULA	

*En el ejemplo de hoja electrónica que aparece en la figura de la izquierda, los datos de las casillas D2, D3 y D4, se habrán definido, respectivamente, por medio de las siguientes fórmulas: (B2*C2), (B3*C3) y (B4*C4). A su vez, como dato de la celda D5 (suma total de los importes) se habrá especificado la fórmula (D2+D3+D4).*
En la figura de la derecha, la pantalla muestra la naturaleza de los distintos elementos de la hoja electrónica utilizada en el ejemplo anterior.

puede surgir la necesidad de hacer operaciones con los datos de distintos resúmenes; éste es el punto en el que entra en escena la hoja electrónica. La forma ideal de acometer estos nuevos estudios es mediante un ordenador personal, al que se le ha hecho llegar la información resumida.

Desde el punto de vista lógico, una hoja electrónica es una gran matriz con filas numeradas (1, 2, 3, ...) y columnas (denominadas A, B, C, ...). De tal forma que cada uno de los elementos de la hoja viene definido por un número que identifica la fila en la que se encuentra, y por una letra, con la que se determina su respectiva columna. El sistema de designación es similar al utilizado en el popular juego de los barcos: una retícula con las casillas identificadas por el cruce de fila y columna.

Cada uno de los elementos de la matriz u hoja electrónica puede ser de tres tipos:

—Literales alfabéticos que servirán únicamente para realizar descripciones de otros elementos de la hoja electrónica.

—Datos numéricos que pueden ser utili-

Elementos de las hojas electrónicas

Al igual que cualquier otro programa o paquete de aplicación, la explotación de una hoja electrónica exige la concurrencia de un determinado número de elementos hardware y software. Elementos que en su expresión mínima se detallan en los siguientes apartados.

• *Elementos hardware*

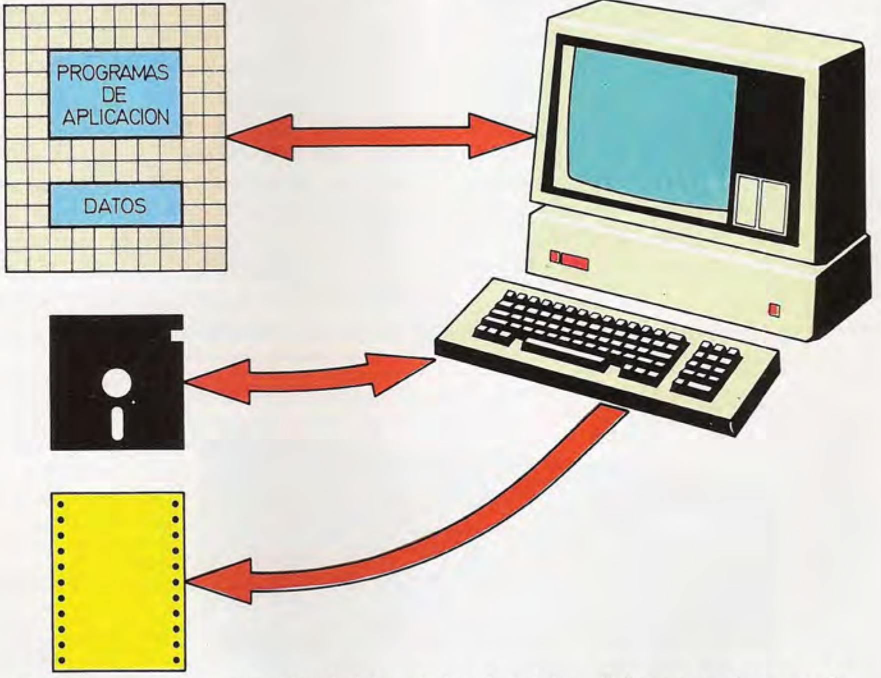
La configuración física del equipo necesario para el empleo de una hoja electrónica, es la que cabe denominar "configuración típica". Esta consta de un procesador o unidad central, una unidad o periférico de entrada/salida y, opcionalmente, alguna unidad de almacenamiento. Como procesador puede servir cualquiera de los que dan naturaleza a un ordenador personal. La unidad de entrada/salida coincide con un terminal o con la pantalla y el teclado propios del ordenador personal; mientras que el periférico de almacenamiento puede ser una unidad de disco o de cinta magnética.

Así como en los tratamientos de textos resultaba de vital importancia la colaboración de una impresora, en el caso de las hojas electrónicas, aun siendo de gran interés, no constituye un elemento imprescindible, puesto que su frecuencia de utilización será netamente inferior.

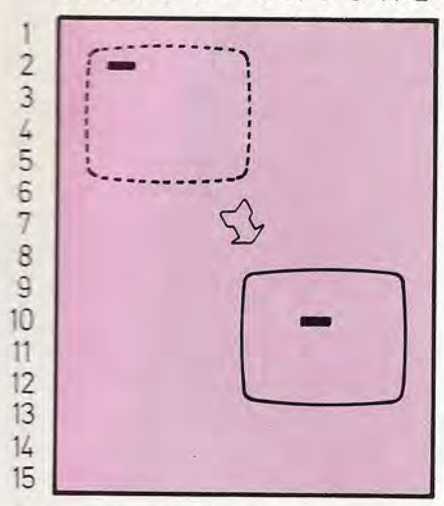
• *Elementos software*

A lo largo de la obra se estudiarán con detalle las principales aplicaciones informáticas de la hoja electrónica. La mayor parte de ellas se basan en programas redactados en lenguaje máquina con objeto de aumentar la velocidad de ejecución. Es fundamental que los programas sean rápidos, ya que cada vez que el usuario realice una modificación en cualquiera de los elementos de la matriz, ello implicará recalcular todos los elementos que contengan fórmulas que, a su vez, incluyan como argumento el elemento modificado.

Obviamente, los programas de hoja electrónica no sólo tienen que gestionar datos sino que, fundamentalmente, deben estar especializados en realizar cálculos de distinto tipo.



Elementos necesarios para el trabajo con hojas electrónicas. Tal como se observa en la figura, éstos se reducen a una configuración hardware —constituida por el ordenador personal, la unidad de almacenamiento (disco) y la impresora—, complementada con el paquete de aplicación al efecto.



Si el tamaño de la hoja electrónica excede a las dimensiones de la pantalla, esta última actuará como una ventana, capaz de desplazarse a través de la superficie de la hoja y visualizar el contenido de cada fragmento.

zados para representar la información numérica manejada.
—Fórmulas de cálculo que manejarán los datos numéricos de la hoja para producir resultados.

De esta forma, cuando el usuario introduce o modifica el contenido de un literal alfabético, el programa se limita a almacenar la descripción introducida. En cambio, cuando se introduce o modifica un elemento numérico, o una fórmula de cálculo, el programa se encargará de recalcular de forma automática todos los elementos relacionados. Evidentemente, tal posibilidad de recálculo convierte a estos programas en poderosas herramientas de trabajo para muy diversos tipos de problemas.

FUNCIONAMIENTO DE UNA HOJA ELECTRONICA

Para poder ejecutar el programa de aplicación se debe preparar el disco flexible, la cinta magnética o cualquier otro soporte físico en el que se encuentre almacenado el programa. A continuación, utilizando el comando de carga del sistema operativo que gobierne al ordenador, hay que introducir el programa en la memoria principal.

Al ejecutarlo, aparecerá en la pantalla una representación en forma de retícula o matriz de casillas, en la que las filas aparecerán numeradas y las columnas se identificarán mediante letras. Para desplazarse a través de los elementos de la matriz, el usuario controlará el movimiento del cursor que, inicialmente, aparecerá situado en el elemento A-1; esto es: en la casilla definida por la intersección de la primera fila y la primera columna.

A continuación, ya sea mediante las teclas de desplazamiento del cursor (arriba, abajo, derecha o izquierda), o bien mediante algún comando del programa, éste puede conseguir que el cursor se posicione en la casilla deseada.

Una vez situado convenientemente el cursor, será preciso teclear el dato que se desee introducir en el elemento correspondiente de la matriz:
—Un literal alfabético.
—Un dato numérico.
—Una fórmula de cálculo.

Estos últimos elementos, una vez definidos, tomarán el valor que resulte de ejecutar las operaciones indicadas en la fórmula a partir de los datos numéricos previamente introducidos. Los cálculos correspondientes a una fórmula no sólo se realizan en el momento de su introducción, sino que serán recalculados por el programa cada vez que se modifique alguno de sus parámetros; desde luego, siempre que el usuario así lo decida.

Existen muchos y muy diversos formatos para la representación en pantalla de los elementos de la matriz que da cuerpo a la hoja electrónica: formato científico, formato entero, formato ajustado a la derecha, formato ajustado a la izquierda, etc. El usuario decide siempre cuál es el formato con que se desea trabajar.

Dado que el tamaño de la pantalla es limitado, tanto para filas como para columnas, el programa permite utilizar la pantalla a modo de ventana: ésta va desplazándose sobre la hoja electrónica y va mostrando distintas porciones de la misma. De esta forma, el movimiento del cursor siempre será relativo al movimiento de la ventana dentro de la hoja.

Mediante la combinación de ambos movimientos (ventana y cursor), el usuario puede examinar todo el contenido de la hoja electrónica, antes de proceder a su escritura en una impresora o a su almacenamiento en un soporte de memoria (cinta, disco, ...). Por supuesto, no es imprescindible terminar una sesión de trabajo con la impresión o almacenamiento de la hoja electrónica. En muchos casos —manteniendo la analogía del programa con el papel, lápiz y máquina de calcular—, los resultados obtenidos sólo servirán para tomar una decisión, y el usuario terminará rompiendo el papel (corresponde a finalizar la ejecución sin almacenamiento ni impresión). En otros casos, una vez terminados los cálculos, interesa preservar el papel con los resultados para archivarlo, utilizarlo o incluso para enviárselo a un tercero (corresponde a finalización sin almacenamiento, pero con impresión). Y, por último, es posible que el usuario quiera conservar la hoja de papel con los datos y resultados para volver a utilizarla en otra ocasión (corresponde a finalización con almacenamiento e impresión).

Prácticamente, todos los fabricantes de ordenadores personales ofrecen, dentro de su catálogo de software de aplicación, una hoja electrónica de cálculo. En los tres próximos capítulos se estudiará detenidamente las características de la primera y tal vez más importante de las hojas electrónicas: VISICALC. Más adelante, en esta misma sección de la obra se repasarán las características técnicas de otros productos informáticos similares.



La unidad de almacenamiento asociada al ordenador personal no sólo está destinada a almacenar el programa de aplicación; también el contenido de las hojas electrónicas elaboradas puede ser memorizado en un soporte externo para su posterior reutilización.

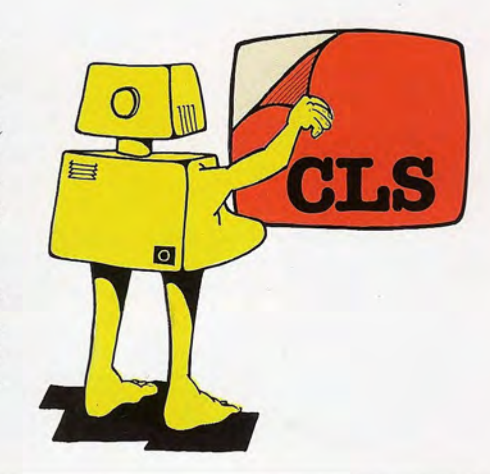
La práctica del BASIC

El programa: un puzzle organizado de comandos e instrucciones

Todo curso de programación ha de tener su zona práctica; sin lugar a dudas "el movimiento se demuestra andando".
En los capítulos anteriores se han introducido los primeros conceptos teóricos del lenguaje BASIC y ha tenido lugar la presentación de un buen número de comandos. Descripciones que se han acompañado de una amplia variedad de ejemplos ilustrativos.
En el presente capítulo, inclinado hacia la vertiente práctica, se van a ensamblar las piezas del puzzle de conceptos, comandos e instrucciones descritos hasta este punto de la obra. Para ello se van a proponer algunos problemas y se va a acometer, paso a paso, su resolución programada.

BORRADO DE LA PANTALLA

Antes de entrar en materia es necesario presentar un nuevo comando BASIC que resultará de inapreciable utilidad a lo largo del capítulo. Se trata del comando CLS (CLear Screen), cuya ejecución se traduce en el borrado del contenido de la pantalla, devolviendo al cursor al principio de la misma.
CLS no incorpora argumento alguno y, normalmente, se puede ejecutar tanto en modo directo como indirecto. Algunos ordenadores disponen incluso de una tecla especial destinada a su introducción inmediata. La mencionada tecla puede llevar la inscripción CLEAR/HOME u otra similar.
Desgraciadamente, el borrado de pantalla no se ajusta a una formulación estándar en la mayoría de los ordenadores. Las alternativas son varias.
Generalmente, los equipos que no disponen de CLS, utilizan una instrucción PRINT para ese mismo cometido. Aquellos que admiten caracteres de control dentro del argumento PRINT, suelen adoptar la siguiente formulación:
PRINT "<tecla de borrado de pantalla>"
A su vez, los ordenadores cuyo dialecto BASIC no admite la inclusión de dichos caracteres, o que no poseen una tecla específica de borrado, suelen utilizar la siguiente instrucción PRINT:
PRINT CHR$ (<num>).







La calidad de la presentación en pantalla condiciona drásticamente el atractivo e incluso la eficacia práctica de un programa. Una muestra palpable la ofrece el simple cartel descriptivo que aparece en la figura.

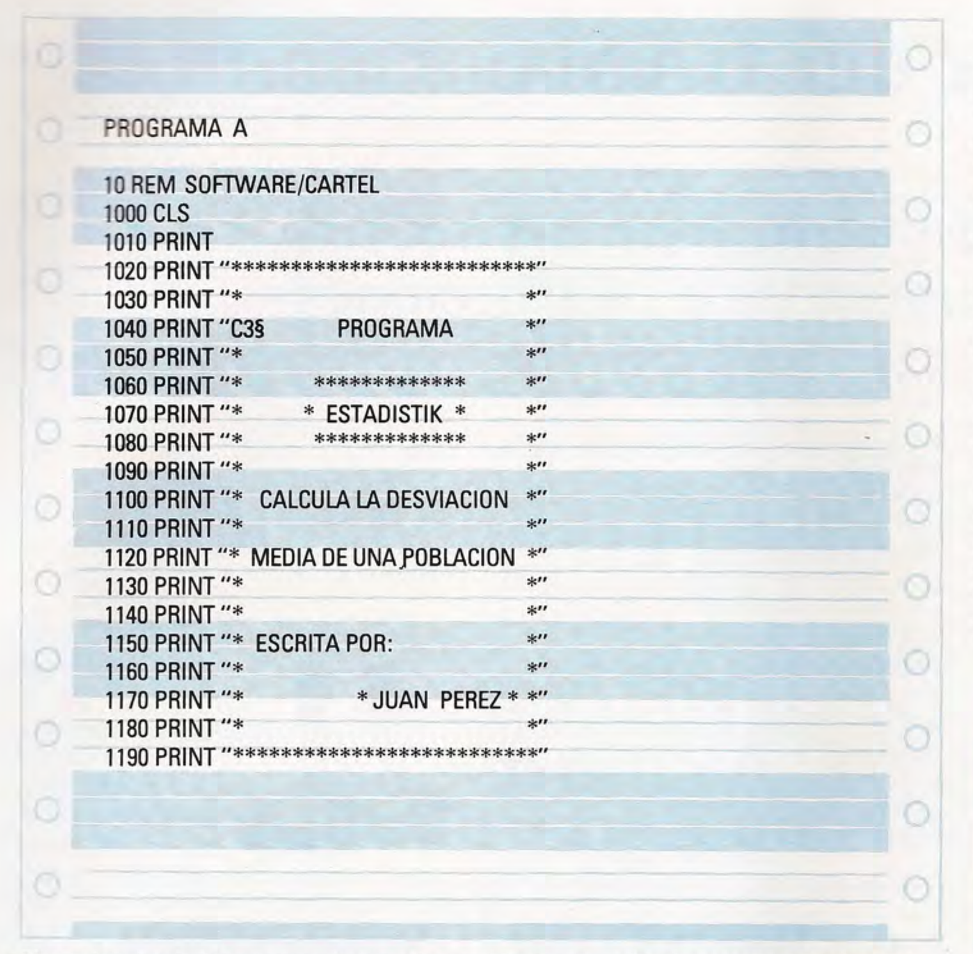
CLS

Borra la pantalla posicionando el cursor en la esquina superior izquierda.

Formato: (Número de línea) CLS

Ejemplos: CLS
20 CLS

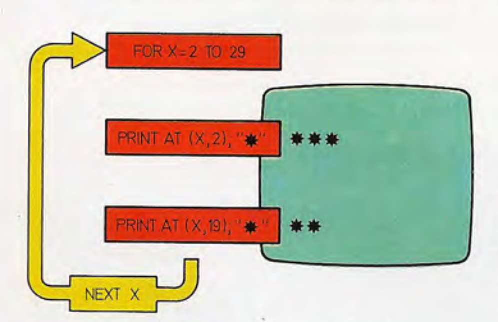
En donde <num> es un número entero específico que coincide con el código de borrado de pantalla (normalmente 26). Por último, existen máquinas que no aportan ninguna instrucción para el borrado de pantalla. En estos casos se hace necesario borrarla "manualmente"; por ejemplo, ejecutando tantos comandos PRINT sin argumento, como líneas tenga la pantalla. Desde luego, el camino apropiado en cada caso conduce a revisar el manual del ordenador para conocer el método que adopta.



PROGRAMA A

```
10 REM SOFTWARE/CARTEL
1000 CLS
1010 PRINT
1020 PRINT "***************************"
1030 PRINT "*                         *"
1040 PRINT "C3$        PROGRAMA       *"
1050 PRINT "*                         *"
1060 PRINT "*     *************       *"
1070 PRINT "*     * ESTADISTIK *      *"
1080 PRINT "*     *************       *"
1090 PRINT "*                         *"
1100 PRINT "*  CALCULA LA DESVIACION  *"
1110 PRINT "*                         *"
1120 PRINT "* MEDIA DE UNA POBLACION  *"
1130 PRINT "*                         *"
1140 PRINT "*                         *"
1150 PRINT "* ESCRITA POR:            *"
1160 PRINT "*                         *"
1170 PRINT "*          * JUAN PEREZ * *"
1180 PRINT "*                         *"
1190 PRINT "***************************"
```

Una primera solución programada para la creación de un cartel en la pantalla del ordenador.



La confección del cartel resultará más cómoda si el usuario recurre a las herramientas que brinda el lenguaje BASIC. Por ejemplo, el trazado de las filas de asteriscos puede encomendarse a un bucle FOR/NEXT, en el que se introducirán sendas instrucciones del tipo PRINT AT.

DISEÑO DE UN CARTEL

Una de las facetas que determinan la calidad de un programa es, sin lugar a dudas, su presentación. Algunos programadores pasan por alto la vertiente estética, prestando exclusivamente su atención a la velocidad de ejecución o a la exactitud de los datos. Desde luego que éste es un objetivo ineludible, si bien, el programa alcanzará una mayor calidad y perfección con una presentación esmerada y agradable. Ello hará incluso que resulte más atractivo y relajado el trabajo con los ordenadores.

La primera herramienta a esgrimir será la versátil instrucción PRINT; imprescindible para la escritura de mensajes en la pantalla.

Como primer ejemplo se diseñará un cartel de entrada a un programa BASIC, en el que aparecerá el nombre del programa y algunos datos de interés. Este constituirá la zona inicial de un programa adecuado para resolver algunos cálculos estadísticos.

A lo largo de este ejercicio se va a trabajar con una pantalla capaz de visualizar 20 filas de 30 caracteres cada una. Por supuesto, en el caso de que el ordenador trabajara con una mayor resolución, habrá que adecuar el ejemplo ampliando las dimensiones del cartel.

La primera alternativa al alcance del programador se reduce a utilizar directamente un bloque de instrucciones PRINT. Por ejemplo:

```
 10 REM SOFTWARE/CARTEL
100 PRINT "PROGRAMA"
110 PRINT "ESTADISTIK"
120 PRINT "CALCULA LA DESVIACION"
130 PRINT "MEDIA DE UNA POBLACION"
140 PRINT "ESCRITO POR:"
150 PRINT "JUAN PEREZ"
■
```

El cartel creado al ejecutar esta rutina resulta muy poco espectacular; aunque tampoco el esfuerzo del programador ha sido excesivo. Su representación en la pantalla es una muestra elocuente de eficacia huérfana de todo condimento estético:

```
RUN

PROGRAMA
ESTADISTIK
CALCULA LA DESVIACION
MEDIA DE UNA POBLACION
ESCRITO POR:
JUAN PEREZ
■
```

Un primer detalle distorsionador lo representa la presencia de la orden RUN en la

pantalla. Esta no forma parte del cartel y, por lo tanto, no debe aparecer. La solución es muy simple y consiste en incluir una línea de borrado de pantalla cuya ejecución se superponga a la palabra RUN. Con ello se eliminará cualquier carácter visualizado en la línea en cuestión.

Otro punto a mejorar es la distribución general del texto. Para darle una mayor simetría, pueden colocarse las frases centradas en la pantalla. Desde luego, también es posible darle una mayor vistosidad encerrando el cartel dentro de un recuadro confeccionado con asteriscos. Su nuevo aspecto es el que refleja la figura adjunta (programa A).

Su presentación resulta bastante más atractiva. De hecho, en el listado aparece el texto distribuido tal y como ha de salir en la pantalla. El único problema radica en el trabajo adicional que ello exige. Mientras que la primera rutina tenía sólo siete líneas, la actual contiene veintiuna (justo el triple). Esto puede hacer que el programador cambie de opinión respecto a la estética del resultado. Pero casi todo tiene solución en el BASIC; veamos hasta qué punto es posible compatibilizar una buena presentación con una programación cómoda.

Una de las zonas de la segunda rutina (programa A) que resultan de más incómoda introducción es el marco de asteriscos. Esta es una tarea perfectamente encomendable al ordenador... ¿No es un verdadero experto en ejecutar tareas reiterativas? Para ello haremos uso de la instrucción FOR/NEXT.

En principio, la línea 1020 del programa A se puede sustituir por el siguiente bloque de instrucciones:

```
1020 PRINT " ";
1021 FOR I=1 TO 28
1022 PRINT "*";
1023 NEXT I
```

Idéntico tratamiento puede darse a la línea 1190. Con lo cual, el usuario se ve liberado de la necesidad de introducir a mano dos filas de asteriscos. El precio a pagar son seis líneas más de programa.

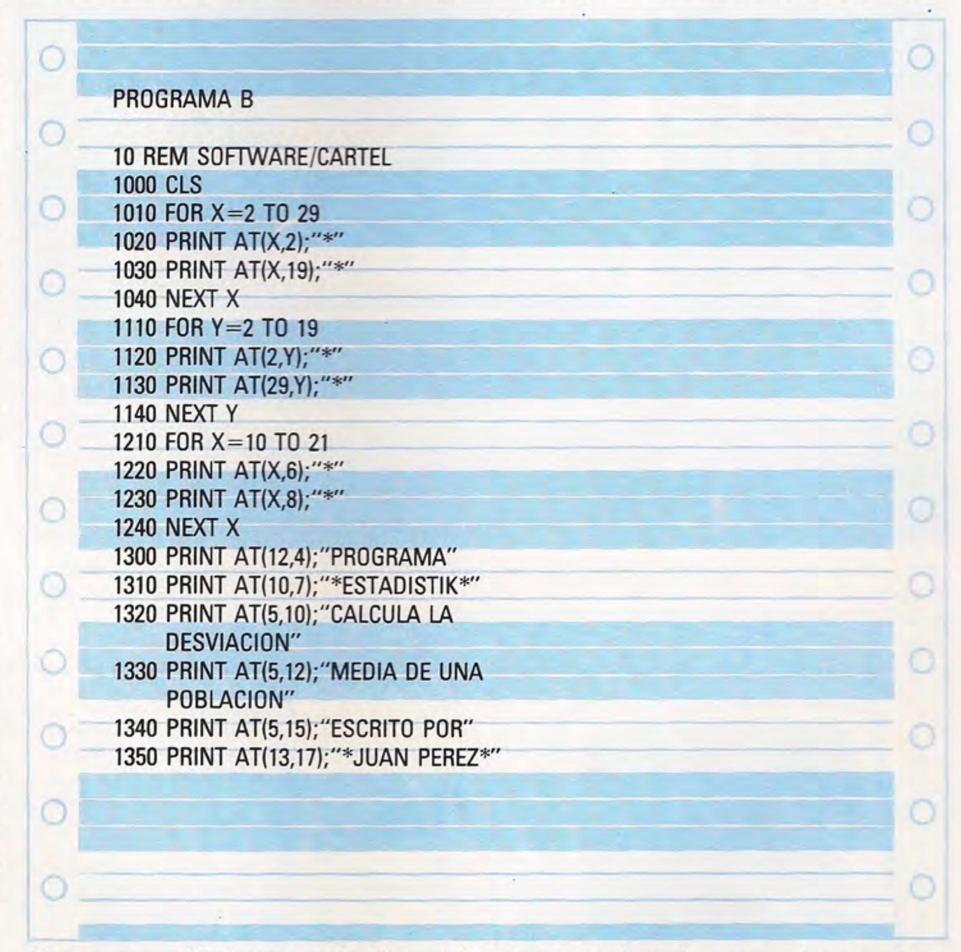
Aún cabe mayor comodidad. Por medio de la variante AT del comando PRINT, se puede colocar el texto sin necesidad de teclear espacios en blanco. Así, por ejemplo, la línea 1040 puede adoptar esta nueva formulación:

```
1040 PRINT AT(12,4);"PROGRAMA"
```

Esta característica del comando PRINT



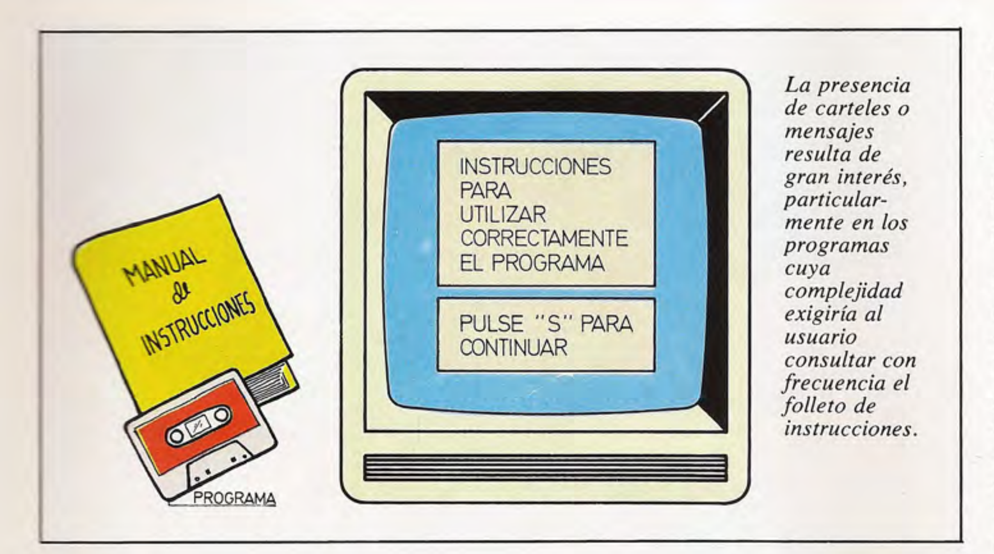
El medio adecuado para ordenar el borrado de la pantalla difiere de uno a otro ordenador. Algunos modelos incorporan el comando CLS, otros recurren a una formulación especial de la instrucción PRINT, e incluso hay equipos que poseen una tecla específica para tal cometido.



```
PROGRAMA B

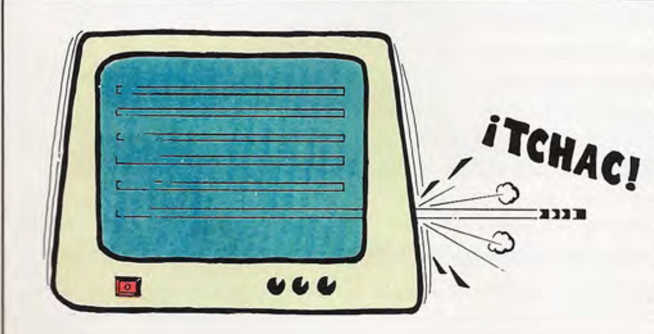
10 REM SOFTWARE/CARTEL
1000 CLS
1010 FOR X=2 TO 29
1020 PRINT AT(X,2);"*"
1030 PRINT AT(X,19);"*"
1040 NEXT X
1110 FOR Y=2 TO 19
1120 PRINT AT(2,Y);"*"
1130 PRINT AT(29,Y);"*"
1140 NEXT Y
1210 FOR X=10 TO 21
1220 PRINT AT(X,6);"*"
1230 PRINT AT(X,8);"*"
1240 NEXT X
1300 PRINT AT(12,4);"PROGRAMA"
1310 PRINT AT(10,7);"*ESTADISTIK*"
1320 PRINT AT(5,10);"CALCULA LA
      DESVIACION"
1330 PRINT AT(5,12);"MEDIA DE UNA
      POBLACION"
1340 PRINT AT(5,15);"ESCRITO POR"
1350 PRINT AT(13,17);"*JUAN PEREZ*"
```

Programa para dibujar un cartel en la pantalla. La nueva variante de programa incorpora las facilidades descritas en el texto. Estas harán que sea el propio ordenador quien se ocupe de dibujar las líneas de cierre, así como de colocar el texto en las zonas previstas de la pantalla.



La presencia de carteles o mensajes resulta de gran interés, particularmente en los programas cuya complejidad exigiría al usuario consultar con frecuencia el folleto de instrucciones.

La calidad de un programa no depende exclusivamente de su precisión, potencia y velocidad de ejecución. También la vertiente estética contribuye a elevar su eficacia, logrando que el uso del programa resulte más cómodo y agradable para el usuario.



Cuando los mensajes son extensos y no caben en una simple pantalla, es preciso recurrir a algún medio que permita detener la visualización en el punto adecuado. De lo contrario, el texto del cartel se desplazará vertiginosamente a través de la pantalla, imposibilitando su lectura.

también permitirá dibujar rápidamente el recuadro; por ejemplo, la fila superior de asteriscos:

```
1020 FOR X=2 TO 29
1021 PRINT AT(X, 2); "*"
1022 PRINT AT(X, 19); "*"
1023 NEXT X
```

El mismo método puede utilizarse para dibujar las dos columnas de asteriscos laterales, e incluso para el trazado del marco interior que encierra el nombre del programa. Por supuesto, en el resto del cartel se seguirá utilizando el comando PRINT AT para situar el texto en el lugar adecuado. Con las nuevas facilidades aportadas por el PRINT AT, se llega a una nueva redacción del programa original. Su aspecto (programa B) es el que muestra la correspondiente figura.

La nueva rutina es bastante más dilatada que la solución original, si bien, no cabe duda que ello ha repercutido en beneficio de la presentación del programa. Y, desde luego, un programa no se puede considerar acabado sin que se haya contemplado la visualización estética de los datos.

UN PEQUEÑO TRUCO

La presencia de los carteles puede resultar muy importante en ciertas ocasiones. Un ejemplo claro lo proporcionan aquellos programas que, por su complejidad, obligan al usuario a conocer las instrucciones o normas de funcionamiento. Estas normas suelen editarse aparte, en el adecuado folleto o manual, aunque también pueden formar parte del programa. Una buena costumbre es incluirlas en el programa. Las normas e instrucciones de uso pueden visualizarse por medio de carteles, semejantes al diseñado. De esta forma, el usuario puede examinarlas mientras se ejecuta el programa y, al tiempo, se evita la distorsión que supondría la pérdida del folleto que las contiene. Cuando es preciso visualizar varios carteles seguidos, surge un problema: ¿cómo detener la ejecución del programa mientras el usuario está leyendo el cartel? De no interrumpir la secuencia de ejecución, las pantallas irían pasando muy rápidamente por la pantalla, imposibilitando su

El código ASCII

Para que el ordenador pueda manipular internamente las letras del alfabeto y los números y signos especiales, es preciso que éstas adopten una forma especial de representación. No hay que perder de vista que los circuitos internos del ordenador sólo pueden trabajar en código binario: con secuencias de ceros y unos. Estos ceros y unos "traducidos" a códigos decimales, aparecerán como números representativos de los distintos caracteres alfabéticos y numéricos. Esta correspondencia hará posible que la máquina pueda manipular correctamente los números y palabras que se introduzcan de acuerdo a dicha codificación.

Evidentemente existen innumerables posibilidades de codificación (tantas como cabe imaginar). Cada usuario puede crear, incluso, su propio código personal para la representación de los caracteres.

Sin lugar a dudas, los códigos personales no resultan útiles en orden a compartir la información representada con otros equipos. De ahí que se intentaran unificar los criterios, con el objetivo de establecer un sistema de representación generalizado. Son varios los códigos para la representación binaria que gozan de aceptación general: por ejemplo, el denominado EBCDIC o el código BAUDOT. Sin embargo, el código más universalmente aceptado es el que responde a las siglas ASCII (American standard code for information interchange: código estándar americano para el intercambio de información).

El código ASCII representa cada carácter por medio de un único byte (ocho bits), de ahí que sus posibilidades de codificación se eleven a 2=256 caracteres distintos. Este número resulta más que suficiente para codificar las letras, cifras y símbolos de uso más frecuente. De ahí que en la práctica acostumbre a reducirse el rango de codificación. Al respecto, la longitud de cada palabra binaria ASCII suele reducirse a siete bits, lo que permite representar a un total de 128 caracteres; número éste más que suficiente en las tareas habituales.

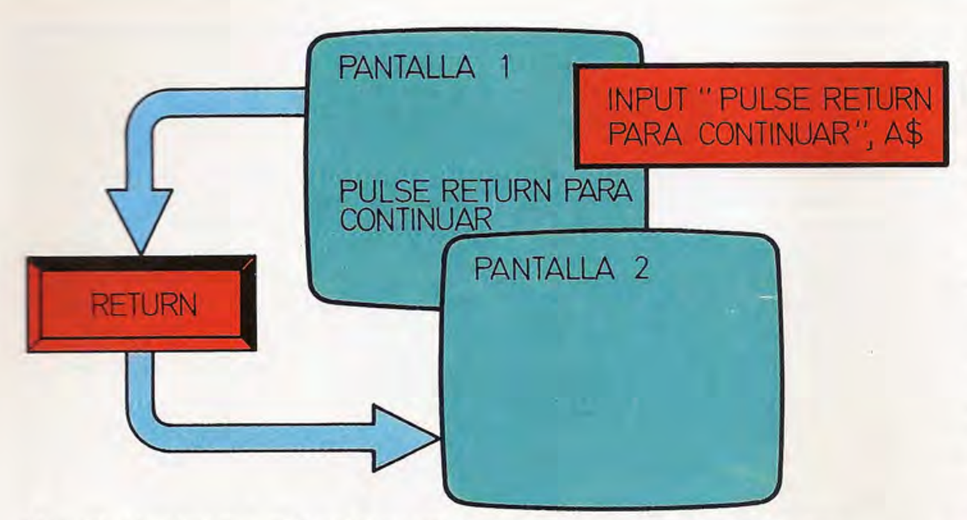
Algunos ordenadores reservan algún cometido específico para el octavo bit libre. Normalmente, lo utilizan para conseguir un juego de caracteres alternativo o un repertorio de caracteres gráficos específicos de la máquina.

En la tabla adjunta aparece relacionado el código ASCII de siete bits. La codificación de cada carácter aparece en expresión decimal (base 10), hexadecimal (base 16), octal (base 8) y binaria. Por ejemplo, el dígito cero tiene asignado el código ASCII 0011000 (48 en expresión decimal) y los nueve siguientes códigos en orden creciente, corresponden a la representación ASCII de las restantes cifras decimales. El abecedario en mayúsculas está ordenado entre los códigos 65 y 90 y el de minúsculas entre los códigos 97 y 122. Los restantes códigos representan caracteres especiales como, por ejemplo: (, $, ?, ", #, ...

CODIGO ASCII

CA-RAC-TER	CODIGO				DEFINICION	CA-RAC-TER	CODIGO				DEFINICION
	BINARIO	DEC.	HEX.	OCT.			BINARIO	DEC.	HEX.	OCT.	
NUL	0000 0000	0	00	000	Nulo	є	0100 0000	64	40	100	"Atpersand"
SOH	0000 0001	1	01	001	Principio de encabezamiento	A	0100 0001	65	41	101	
STX	0000 0010	2	02	002	Comienzo de texto	B	0100 0010	66	42	102	
ETX	0000 0011	3	03	003	Fin de texto	C	0100 0011	67	43	103	
EOT	0000 0100	4	04	004	Fin de transmisión	D	0100 0100	68	44	104	
ENQ	0000 0101	5	05	005	Pregunta	E	0100 0101	69	45	105	
ACK	0000 0110	6	06	006	Acuse de recibo	F	0100 0110	70	46	106	
BEL	0000 0111	7	07	007	Timbre (señal)	G	0100 0111	71	47	107	
BS	0000 1000	8	08	010	Retroceso	H	0100 1000	72	48	110	
HT	0000 1001	9	09	011	Tabulación horizontal	I	0100 1001	73	49	111	
LF	0000 1010	10	0A	012	Cambio de renglón	J	0100 1010	74	4A	112	
VT	0000 1011	11	0B	013	Tabulación horizontal	K	0100 1011	75	4B	113	
FF	0000 1100	12	0C	014	Página siguiente	L	0100 1100	76	4C	114	
CR	0000 1101	13	0D	015	Retroceso de carro	M	0100 1101	77	4D	115	
SO	0000 1110	14	0E	016	Fuera de código	N	0100 1110	78	4E	116	
SI	0000 1111	15	0F	017	En código	O	0100 1111	79	4F	117	
DLE	0001 0000	16	10	020	Encaje de transmisión	P	0101 0000	80	50	120	
DC1	0001 0001	17	11	021	Mando de dispositivo auxiliar 1	Q	0101 0001	81	51	121	
DC2	0001 0010	18	12	022	Mando de dispositivo auxiliar 2	R	0101 0010	82	52	122	
DC3	0001 0011	19	13	023	Mando de dispositivo auxiliar 3	S	0101 0011	83	53	123	
DC4	0001 0100	20	14	024	Mando de dispositivo auxiliar 4	T	0101 0100	84	54	124	
NAK	0001 0101	21	15	025	Acuse de recibo negativo	U	0101 0101	85	55	125	
SYN	0001 0110	22	16	026	Sincronización de reposo	V	0101 0110	86	56	126	
ETB	0001 0111	23	17	027	Fin de bloque de transmisión	W	0101 0111	87	57	127	
CAN	0001 1000	24	18	030	Cancelación	X	0101 1000	88	58	130	
EM	0001 1001	25	19	031	Fin de medio físico	Y	0101 1001	89	59	131	
SUB	0001 1010	26	1A	032	Sustitución	Z	0101 1010	90	5A	132	
ESC	0001 1011	27	1B	033	Escape	[	0101 1011	91	5B	133	Apertura de corchete
FS	0001 1100	28	1C	034	Separador de fichero	\	0101 1100	92	5C	134	Barra invertida ("Back slash")
GS	0001 1101	29	1D	035	Separador de grupo	]	0101 1101	93	5D	135	Cierre de corchete
RS	0001 1110	30	1E	036	Separador de registro	^	0101 1110	94	5E	136	Acento circunflejo
US	0001 1111	31	1F	037	Separador de unidad	_	0101 1111	95	5F	137	Guión de subrayado
	0010 0000	32	20	040	Espacio en blanco	`	0110 0000	96	60	140	Acento inverso
!	0010 0001	33	21	041	Admiración	a	0110 0001	97	61	141	
"	0010 0010	34	22	042	Comillas	b	0110 0010	98	62	142	
#	0010 0011	35	23	043	Símbolo número	c	0110 0011	99	63	143	
$	0010 0100	36	24	044	Símbolo dólar	d	0110 0100	100	64	144	
%	0010 0101	37	25	045	Porcentaje	e	0110 0101	101	65	145	
&	0010 0110	38	26	046	"Ampersand"	f	0110 0110	102	66	146	
'	0010 0111	39	27	047	Acento	g	0110 0111	103	67	147	
(	0010 1000	40	28	050	Apertura de paréntesis	h	0110 1000	104	68	150	
)	0010 1001	41	29	051	Cierre de paréntesis	i	0110 1001	105	69	151	
*	0010 1010	42	2A	052	Asterisco	j	0110 1010	106	6A	152	
+	0010 1011	43	2B	053	Signo más	k	0110 1011	107	6B	153	
,	0010 1100	44	2C	054	Coma	l	0110 1100	108	6C	154	
–	0010 1101	45	2D	055	Guión (signo menos)	m	0110 1101	109	6D	155	
.	0010 1110	46	2E	056	Punto	n	0110 1110	110	6E	156	
/	0010 1111	47	2F	057	Símbolo división ("slash")	o	0110 1111	111	6F	157	
0	0011 0000	48	30	060		p	0111 0000	112	70	160	
1	0011 0001	49	31	061		q	0111 0001	113	71	161	
2	0011 0010	50	32	062		r	0111 0010	114	72	162	
3	0011 0011	51	33	063		s	0111 0011	115	73	163	
4	0011 0100	52	34	064		t	0111 0100	116	74	164	
5	0011 0101	53	35	065		u	0111 0101	117	75	165	
6	0011 0110	54	36	066		v	0111 0110	118	76	166	
7	0011 0111	55	37	067		w	0111 0111	119	77	167	
8	0011 1000	56	38	070		x	0111 1000	120	78	170	
9	0011 1001	57	39	071		y	0111 1001	121	79	171	
:	0011 1010	58	3A	072	Dos puntos	z	0111 1010	122	7A	172	
;	0011 1011	59	3B	073	Punto y coma	<	0111 1011	123	7B	173	Apertura de corchete
<	0011 1100	60	3C	074	Menor que	|	0111 1100	124	7C	174	Barra vertical
=	0011 1101	61	3D	075	Igual	>	0111 1101	125	7D	175	Cierre de corchete
>	0011 1110	62	3E	076	Mayor que	~	0111 1110	126	7E	176	Tilde
?	0011 1111	63	3F	077	Interrogante	DEL	0111 1111	127	7F	177	Borrado, supresión

DEC.=Decimal. HEX.=Hexadecimal. OCT.=Octal.



Una solución cómoda y útil para fragmentar la visualización de carteles en sucesivas pantallas, la constituye el empleo de un "falso INPUT". Al ejecutarlo, el ordenador detiene la visualización, presentando el mensaje asociado y aguardando a que el usuario accione alguna tecla para pasar a la siguiente pantalla.

lectura. Una posible solución la aporta el comando STOP; en efecto, su especialidad no es otra que detener la ejecución. En tal caso, bastaría con introducir un comando CONT para reanudarla. Para evitar la necesidad de teclear tantas letras (cuatro de la palabra CONT más la tecla RETURN) existen varios "trucos". Por ejemplo, el BASIC dispone de algunas funciones que permiten detectar una pulsación en el teclado, pero esta solución es aún

El lenguaje BASIC ofrece herramientas adecuadas para el trabajo en la pantalla del ordenador. Herramientas que permiten visualizar mensajes y construir presentaciones a voluntad del programador.

prematura, ya que de ellas se hablará en posteriores capítulos.

Los conocimientos adquiridos en capítulos anteriores son suficientes para dar con una solución. El truco consiste en recurrir a un "falso INPUT". Cabe recordar que dicha instrucción detiene la ejecución del programa hasta que se introduce un dato a través del teclado. Así pues, bastará con pedir al usuario que introduzca un dato cuando haya terminado de leer el cartel. Esta solución no resulta excesivamente elegante, ya que la introducción de un dato cuando es irrelevante, puede dar lugar a confusiones. Por lo tanto, enmascararemos al INPUT, pidiendo tan sólo la pulsación de la tecla RETURN. Esta sería, en definitiva, la línea a incluir tras las que corresponden a la visualización del cartel:

```
1400 INPUT "PULSE RETURN PARA
     CONTINUAR",A$
```

Su presencia hará que el ordenador interrumpa el programa, cuya ejecución reanudará cuando el usuario pulse la tecla RETURN.

La coma localizada detrás del mensaje evita que la pantalla muestre el signo de interrogación característico de INPUT. Al pulsar la tecla RETURN sin haber introducido previamente dato alguno, la variable A$ no recibirá la asignación de valor alguno y, en consecuencia, quedará "vacía". Ello no supone ningún problema, sino que basta, simplemente, con ignorar dicha variable.

Si se desea situar el mensaje en un punto determinado de la pantalla, habrá que utilizar previamente una instrucción de tipo PRINT AT. En nuestro ejemplo será suficiente con añadir las dos líneas siguientes para que quede contemplada la referida posibilidad:

```
1340 PRINT AT(2, 20);
1400 INPUT "PULSE RETURN PARA
     CONTINUAR",A$
```

La línea 1390 obliga a que el mensaje que acompaña a INPUT se imprima a partir de la posición señalada por el argumento de AT. Con ello, es posible colocar el mensaje en el punto deseado y no en la línea siguiente a la utilizada por el último PRINT.

DE LA ESTADISTICA AL JUEGO

El ordenador representa una gran ayuda para el trabajo y la investigación dado su gran poder de cálculo. Su capacidad matemática también puede utilizarse para otra finalidad más lúdica; para el ocio, por ejemplo. A través de este apartado van a conjugarse las posibilidades matemáticas y lúdicas del ordenador de la mano de un programa.

El juego consiste en simular el lanzamiento de un proyectil, con la precisión adecuada para acertar en el blanco. Para ello, se suministrarán a la máquina la velocidad inicial y el ángulo de disparo.

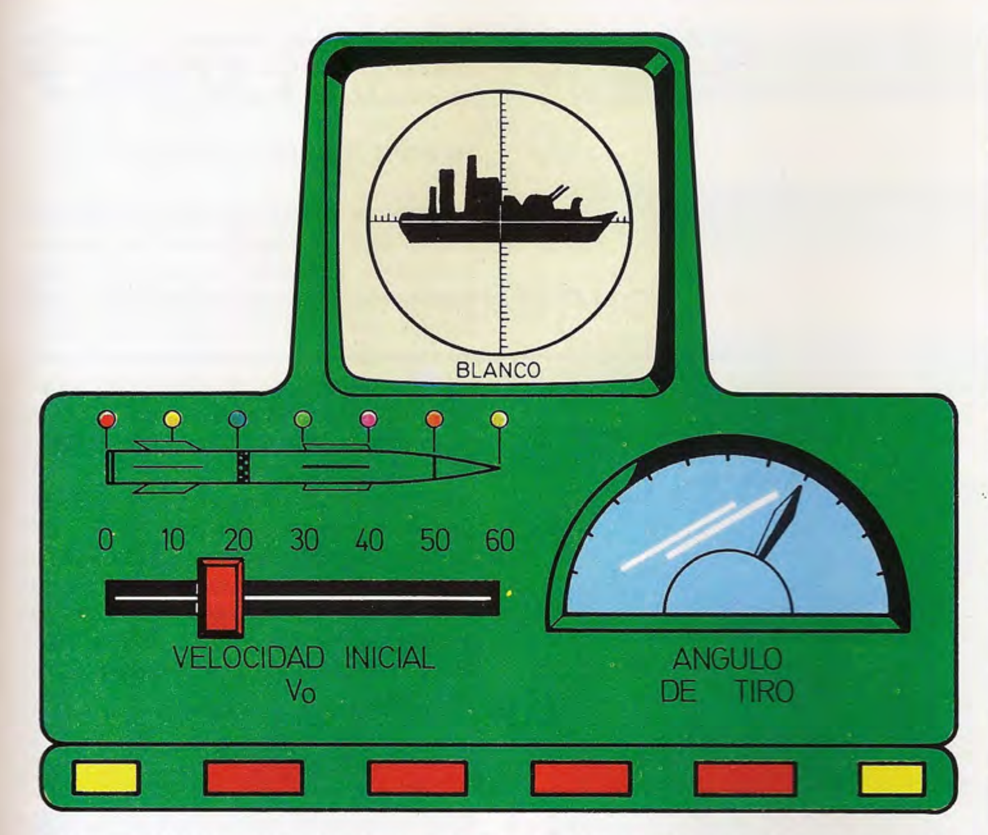
El programa se reduce a la resolución matemática de un tiro parabólico. La ecuación de partida es la siguiente:

```
S=V0*T+(1/2)*A*t↑2
```

La velocidad inicial (V0) se separa en sus dos componentes según las direcciones vertical y horizontal. Para ello, se hará uso de los datos iniciales (velocidad "V0" y ángulo "ANG") y de las dos ecuaciones que siguen:

```
V0x=V0*COS(ANG)
V0y=V0*SEN(ANG)
```

Los referidos datos permiten calcular la distancia alcanzada por el proyectil sin más que introducirlos en la ecuación inicial:



Escenario del juego creado por el programa "Tiro parabólico". El jugador debe acertar en el blanco, seleccionando la velocidad inicial del proyectil y el ángulo de tiro con la precisión adecuada.

TABLA DE CONVERSION

ORDENADOR	CLS
	CLS
APPLE II (APPLESOFT)	CALL-936
APRICOT (M-BASIC)	PRINT CHR$ (26)
ATARI	PRINT " ↰ "
CBM 64	PRINT "[♥]"
DRAGON	CLS
EQUIPOS MSX	CLS
HP-150	—
IBM PC	CLS
MPF	HOME
NCR DM-V (MS-BASIC)	PRINT CHR$ (26)
NEW BRAIN	—
ORIC	CLS
SHARP MZ-700 (MZ-BASIC)	PRINT "[C]" PRINT CHR$ (6)
SINCLAIR QL	CLS
SPECTRAVIDEO	CLS
ZX-SPECTRUM	CLS

```
0=V0y-(1/2)*G*T
Sx=V0x*T
```

(G es la aceleración de la gravedad, cuyo valor es de 9,81 m/s).

Con estos conocimientos se está ya en disposición de crear el cuerpo o zona principal del programa.

```
200 LET G=9,81
210 LET VX=V0*COS(ANG)
220 LET VY=V0*SIN(ANG)
230 LET T=(VY*2)/G
240 LET SX=VX*T
```

La tarea realizada ha sido convertir las ecuaciones matemáticas a líneas de programa en lenguaje BASIC. Pero estas líneas no funcionan por sí solas, sino que es necesario introducir los datos iniciales: velocidad inicial y ángulo del proyectil. Esto se logra por medio de un par de instrucciones INPUT:

```
110 INPUT "VELOCIDAD INICIAL";V0
120 INPUT "ANGULO DE TIRO";ANG
```

Cualquier ordenador suele trabajar, normalmente, con ángulos expresados en radianes. Por lo tanto, el dato a introducir en la línea 120 debe ser el número de radianes deseado. Si al usuario le resulta más cómodo utilizar grados, es necesario que teclee estas otras dos líneas, adecuadas para la conversión de grados a radianes:

```
130 LET PI=3.1416
140 LET ANG=(ANG*PI)/180
```

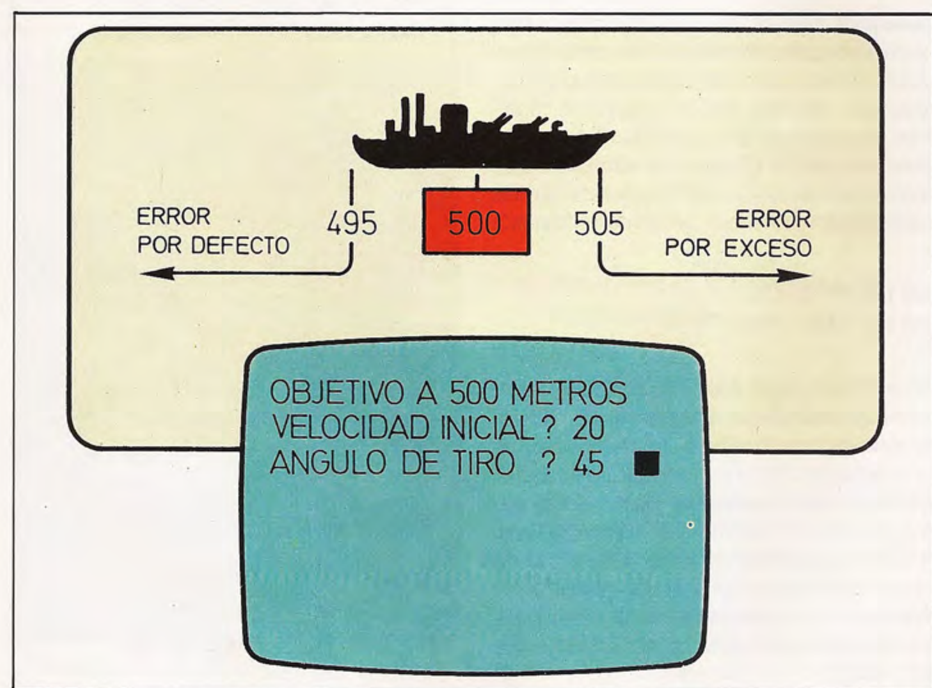
El próximo paso consiste en detectar si se ha producido un impacto en el blanco. En primer lugar hay que situar el blanco en una posición fija. Por ejemplo, a una distancia de 500 metros del punto del lanzamiento del proyectil. Se supondrá, asimismo, que el blanco mide 10 metros de largo. Ello significa que se considerará acertado si la distancia SX a la que impacta el proyectil está comprendida entre 495 y 505.
Si al calcular SX se obtiene un valor superior a 505, se habrá errado el tiro por exceso. En tal caso, el ordenador debe indicar a qué distancia del blanco cayó el proyectil. Algo parecido ocurre cuando SX toma un valor inferior a 495, con la salvedad de que el error en el tiro será por defecto.
Una vez programada, la zona para la detección de la eficacia del tiro adoptará el siguiente aspecto.

```
300 IF SX>505 THEN GOTO 400
310 IF SX<495 THEN GOTO 500
320 PRINT "BLANCO"
330 END
400 PRINT "DISTANCIA=";SX
410 PRINT "FALLO POR";SX-500
420 GOTO 100
500 PRINT "DISTANCIA=";SX
510 PRINT "FALLO POR";500-SX
520 GOTO 100
```



```
PROGRAMA C

10 REM TIRO PARABOLICO
20 FOR I=1 TO 5
100 PRINT "OBJETIVO A 500 METROS"
110 INPUT "VELOCIDAD INICIAL";VO
120 INPUT "ANGULO DE TIRO"; ANG
130 LET PI=3.1416
140 LET ANG=(ANG*PI)/180
200 LET G=9.81
210 LET VX=VO*COS(ANG)
220 LET VY=VO*SIN(ANG)
230 LET T=(VY*2)/G
240 LET SX=VX*T
300 IF SX>505 THEN GOTO 400
310 IF SX<495 THEN GOTO 500
320 PRINT "BLANCO"
330 END
400 PRINT "DISTANCIA=";SX
410 PRINT "FALLO POR"; SX-500;
    "METROS"
420 GOTO 600
500 PRINT "DISTANCIA=";SX
510 PRINT "FALLO POR"; 500-SX;
    "METROS"
600 NEXT I
610 PRINT "NO HA CONSEGUIDO"
620 PRINT "DESTRUIR EL OBJETIVO"
700 END
```



El programa considera que la longitud del blanco es de diez metros. En consecuencia, resultará eficaz cualquier disparo cuyo impacto se produzca a una distancia de 495 a 505 metros del origen del proyectil.

En la línea 100 es conveniente colocar un mensaje que indique la distancia a la que se encuentra el objetivo.

```
100 PRINT "OBJETIVO A 500 METROS"
```

Por último, y para dar mayor emoción al juego, se limitará el número de disparos realizables. Para ello, se introducirá la rutina dentro de un bucle FOR/NEXT. La variable del bucle (I) irá contabilizando el número de disparos efectuados. En el ejemplo, se dispone de cinco oportunidades (límite de la variable FOR).
El programa, completado con la incorporación de las últimas condiciones enunciadas, coincide con el que aparece en la figura adjunta (programa C).
La ejecución del programa se traduce en la presentación en pantalla que aparece a continuación. Los valores elegidos son 20 metros por segundo para la velocidad inicial y un ángulo de salida del proyectil de 25 grados.

```
RUN

OBJETIVO A 500 METROS
VELOCIDAD INICIAL? 20
ANGULO DE TIRO? 25
DISTANCIA=31.235247
FALLO POR 468.76475 METROS
OBJETIVO A 500 METROS
VELOCIDAD INICIAL? ■
```

La petición de datos por parte del ordenador se repetirá en cinco ocasiones; por supuesto, siempre que el disparo no logre dar en el blanco, en cuyo caso se dará por terminado el programa en la línea 320.
Si el jugador no consigue abatir al blanco enemigo, el programa finalizará mostrando en la pantalla el siguiente mensaje:

```
OBJETIVO A 500 METROS
VELOCIDAD INICIAL? 75
ANGULO DE TIRO? 45
DISTANCIA=573.3945
FALLO POR 73.394495 METROS
NO HA CONSEGUIDO
DESTRUIR EL OBJETIVO
■
```

Logo (9)

TURTLE GRAPHICS: la tortuga se multiplica

A pesar de la gran diversidad de posibilidades estudiadas hasta el momento, el método "turtle graphics" sigue encerrando nuevas funciones aún por desvelar. Funciones que permiten un mayor control sobre la tortuga, ya sea para establecer su velocidad de desplazamiento, o para multiplicar su presencia en la pantalla.

CONTROL DE LA VELOCIDAD

Una de las características de la tortuga hasta ahora ignorada, es la posibilidad de alterar su velocidad de movimiento. Nuetro disciplinado personaje no tiene por qué permanecer inmóvil, sino que puede lanzarse a recorrer la pantalla con una determinada velocidad. El comando implicado es SETSP (SET SPeed).

Por ejemplo, SETSP 20 hará que la tortuga camine con una velocidad de 20. Desde luego, su paseo no cesará hasta que se le asigne una velocidad cero. Si la tortuga se encuentra con la tiza "bajada", ésta irá dibujando su trayectoria sobre la pantalla.

La orden SETSP no altera las restantes condiciones definidas, sino que mantiene los atributos correspondientes a posición, número y color de la tiza. Cabe notar que la velocidad determina un movimiento rectilíneo en la dirección apuntada por el eje longitudinal de la tortuga.

Es necesario significar las distintas reacciones de la tortuga dependiendo del modo en el que estén definidos los límites de la pantalla. Por ejemplo, si se ha seleccionado el modo cerrado (con la orden WRAP), se verá a la tortuga aparecer por el punto opuesto a aquél por donde ha abandonado la pantalla. En el caso de que la tortuga tuviera una orientación oblicua con respecto a los ejes de coordenadas, ésta no correría siempre la misma línea diagonal.

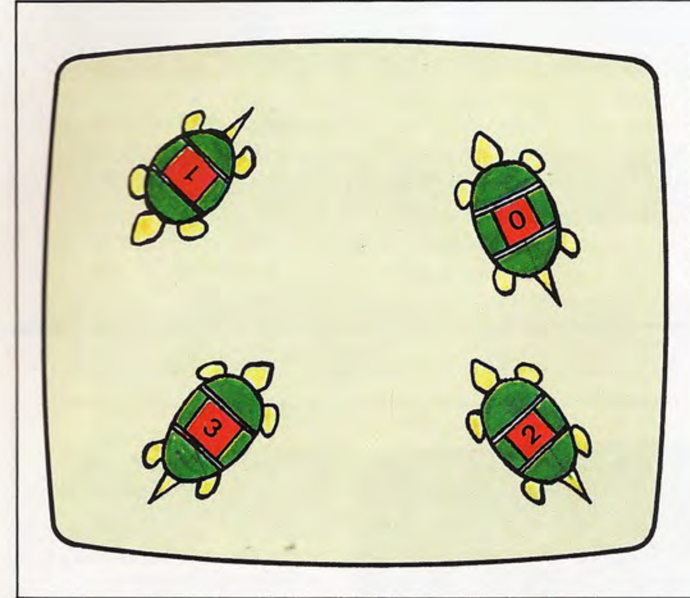
Cuando se opta por la pantalla de límites abiertos (WINDOW), la tortuga desaparece del campo visual y continúa avanzando hasta llegar al final del terreno disponible. La extensión de este terreno depende de la capacidad propia del ordenador. Asimismo, las velocidades máxima y mínima admisibles vienen impuestas por las características del equipo.

Como ya es habitual, el LOGO dispone de la función opuesta. El operador SPEED es el encargado de devolver el indicativo de la velocidad actual. El siguiente procedimiento muestra la acción de las dos órdenes presentadas.

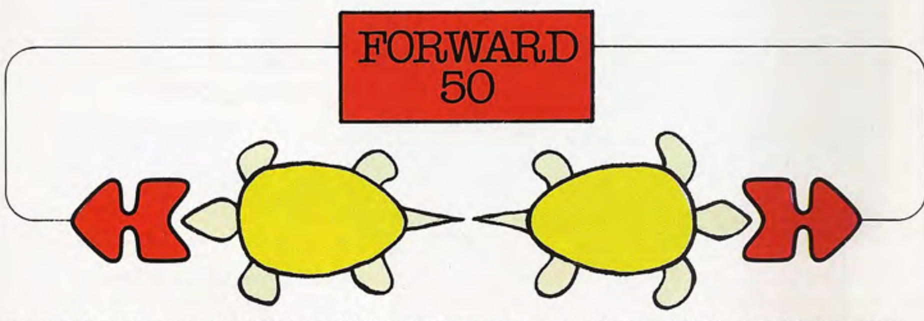
El comando SETSP hace que la tortuga se mantenga en movimiento a la velocidad especificada. El operador complementario es SPEED; su ejecución devuelve el valor de la velocidad de desplazamiento.

```
TO EJEMPLO
WRAP
MAKE "A [ACELERA]
CORRE
END
```

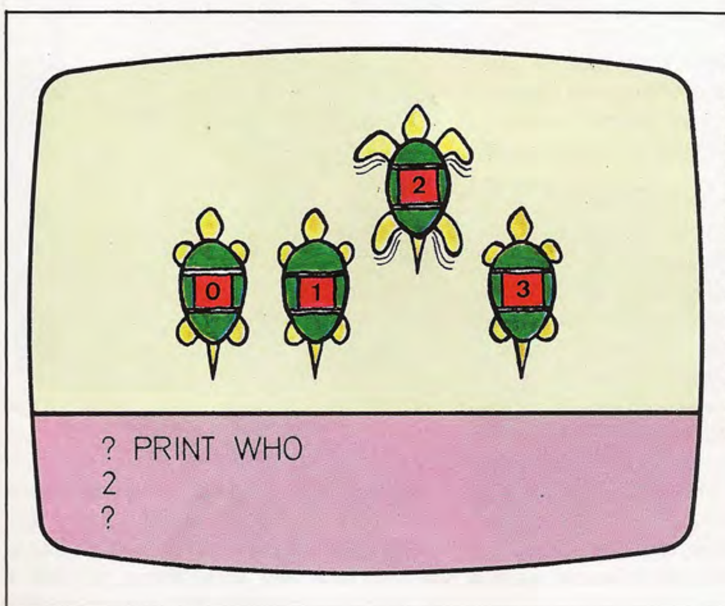
Como se observa, el procedimiento EJEMPLO incluye en su definición a dos procedimientos elementales, ACELERA y CORRE; a su vez, éste último engloba a un tercer procedimiento elemental: DE-



El LOGO permite el uso simultáneo de varias tortugas. Habitualmente, son cuatro las tortugas utilizables; para permitir su control, éstas se identifican por medio de un número comprendido entre cero y tres.



La misión del comando TELL es definir qué tortugas van actuar a partir de ese instante. Tras llamar a dos tortugas —localizadas en distintos puntos de la pantalla o con distinta orientación— por medio del mencionado comando y ordenar un determinado movimiento, éstas ejecutarán el desplazamiento por distinta zona de la pantalla, de acuerdo a su posición y orientación de partida.



WHO es un operador que revela el número de la tortuga "en curso" o tortuga llamada en última instancia. Si son varias las tortugas activas en el instante de hacer uso del mencionado operador, el ordenador mostrará los números de todas ellas.

CELERA. Todos ellos aparecen definidos a continuación.

```
TO CORRE
IF SPEED>100 [MAKE"A [DECELERA]
RUN :A
IF SPEED<1 [STOP]
END
```

```
TO ACELERA
SETSP SPEED+2
END

TO DECELERA
SETSP SPEED-2
END
```

La ejecución del procedimiento EJEMPLO hará que la velocidad de la tortuga aumente de 0 a 100, en pasos de 2 unidades. Al llegar a una velocidad mayor que 100, la tortuga empezará a decelerar, hasta que su velocidad vuelva a ser cero. Las instrucciones RUN e IF utilizadas en el procedimiento CORRE se describirán en un capítulo posterior dedicado al estudio de los bucles.

DIBUJANDO CON VARIAS TORTUGAS

Una sorpresa agradable la constituye el hecho de que la tortuga pueda también actuar en equipo. El LOGO permite trabajar simultáneamente con más de una tortuga en escena.
Habitualmente, el número máximo de tortugas es de cuatro, identificadas con los números del cero al tres. La tortuga con la que se ha trabajado hasta ahora es la número cero. Para que entren en juego las demás, es preciso "llamarlas" por medio del comando TELL (decir, "dile a" ...). Este comando se emplea también para comunicar una orden a una tortuga específica.

Por ejemplo, TELL 1 hace aparecer a la tortuga número 1; a su vez, las órdenes que sigan serán obedecidas por la última tortuga invocada (en este caso, la número 1). Si se pretende que las órdenes afecten simultáneamente a varias tortugas, habrá que advertir de ello a los quelonios. Por ejemplo, para que las tortugas "actuantes" sean las dos primeras, habrá que empezar tecleando TELL [0 1]. En general, es preciso formar una lista con los números de las tortugas que deban obedecer al tiempo las mismas órdenes, y llamarlas con un comando TELL.
Las últimas tortugas invocadas con TELL se denominan *tortugas en curso,* por ser ellas las que obedecen las órdenes posteriores. Si se invoca a varias tortugas situadas en distintas posiciones de la pantalla, lo que se les mande dibujar lo ejecutarán todas ellas a la vez, aunque cada una en la zona que le corresponda. Cambiando de tortuga en curso, es posible controlar a cada una de ellas por separado.
Para identificar a la tortuga o tortugas que están actuando en un determinado momento, hay que hacer uso del operador WHO (quién). Su ejecución responde con el número de la tortuga o tortugas invocadas en último lugar. Este operador resulta útil para dar órdenes distintas a cada tortuga. Por ejemplo, una instrucción del tipo FORWARD WHO*10, hará que si la tortuga en curso es la uno, ésta avance 10 posiciones, si es la dos, 20 ...

TORTUGAS BAJO CONTROL

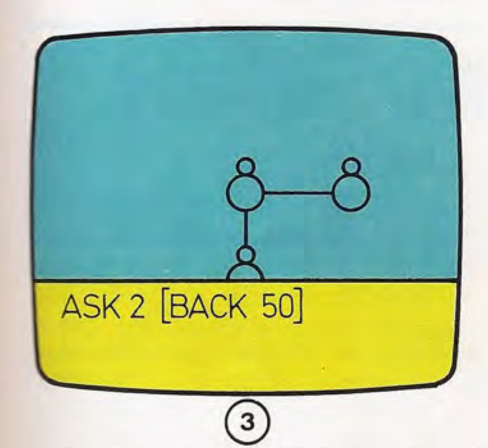
Para dirigir a cada tortuga por separado se hace necesario llamarlas alternativamente. Ello se puede conseguir con el uso reiterado del comando TELL, o escribiendo procedimientos que utilicen dicho comando.
El siguiente ejemplo cambia la tortuga actuante por la identificada por el siguiente número.

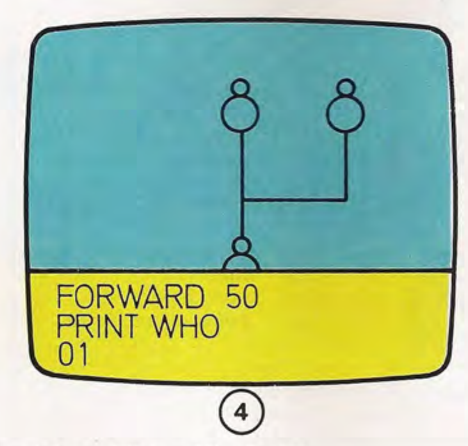
```
TO CAMBIATOR
IF WHO=3 [TELL 0] [TELL WHO+1]
END
```

Cada vez que se ejecute el procedimiento cambiará la tortuga en curso. Con éste y otros procedimientos semejantes, el programador estará en condiciones de alterar la situación de las tortugas, para que cada una realice una tarea distinta. Existen también otros comandos que permiten una mayor flexibilidad en el uso de varias tortugas. Estos son los que se detallan a continuación.
Cuando las órdenes a ejecutar por cada tortuga sean idénticas, se puede utilizar EACH (cada). La sintaxis de este comando es la siguiente: EACH <lista de instrucciones>. Por ejemplo, EACH [FORWARD 200] hará que *cada* tortuga avance 200 posiciones. La diferencia con la situación normal consiste en que, ahora, la segunda tortuga no cursa la orden hasta que termina de hacerlo la primera. De no utilizar el comando EACH, todas las tortugas "acatarían" la orden a la vez.
La principal utilidad de EACH radica en su empleo conjunto con WHO. Tal combinación permitirá ejecutar órdenes distintas a cada tortuga; por supuesto, siempre que las órdenes se puedan reducir a una función del número de cada tortuga.
Por ejemplo, para que cada tortuga gire en un ángulo de distinto número de grados, se puede introducir la orden: EACH [RIGHT WHO*90]. Con el ejemplo apuntado más arriba, en el que se utilizaba WHO sin EACH, se producirá un error si hay más de una tortuga en acción, simultáneamente. Ello se debe a que WHO devuelve una lista de números. Sin embargo, cuando actúa en compañía de EACH, el LOGO se encarga de extraer de la lista cada número por separado.
EACH no cambia la tortuga en curso y sólo actúa con las que estén disponibles en ese instante.
El método adecuado para emplear una tortuga que no sea la actualmente en curso, sin por ello cambiarla, llega de la

1
TELL [0 1]
PRINT WHO
01

2
EACH [SET X WHO * 50]





Secuencia de pantallas que ilustra el efecto de las órdenes TELL, ASK y EACH.

mano de ASK. Este comando exige dos datos de entrada: el primero es el número de tortuga y el segundo la lista de las instrucciones que ha de "obedecer". Si se desea que las instrucciones las ejecuten varias tortugas, el primer dato debe ser una lista con los números de las tortugas afectadas.
El siguiente ejemplo ilustra la actuación de cada comando.

```
TELL [0 1]
PRINT WHO
0 1

EACH [SETX WHO *50]
ASK 2 [BACK 50]
PRINT WHO
0 1
```

Como se observa, ASK no altera las tortugas en curso.
Otra posibilidad de ASK radica en su empleo como operador. Realmente, la acción de ASK es la que defina la lista de instrucciones que constituyen su dato de entrada. Ello significa que se comportará como un operador si la lista da un dato de salida. He aquí un ejemplo:
PRINT ASK 1 [POS].
En este caso, las coordenadas de la tortuga 1 constituyen el dato de salida, a raíz de lo cual, ASK actúa como operador.

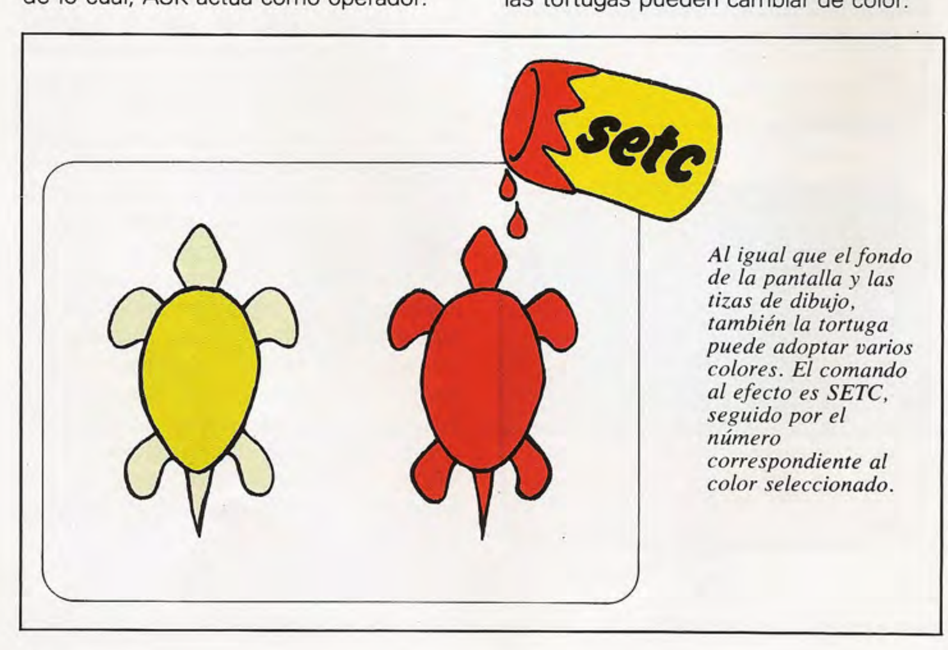
TABLA DE ORDENES DEL "TURTLE GRAPHICS"

Instrucción	Cometido	Operador/comando
SETSP<número>	Hace que la tortuga se mantenga en movimiento a la velocidad dada por <número>	Comando
SPEED	Devuelve la velocidad actual de la tortuga	Operador
TELL<nu/lis>	Define qué tortugas serán las actuantes a partir de ese momento	Comando
WHO	Devuelve una lista con los números de las tortugas en curso	Operador
EACH...<lista ins>	Hace que las tortugas en curso ejecuten la lista de instrucciones, una tortuga después de otra	Comando
ASK<nu/lis> <lista ins>	Hace que la o las tortugas especificadas ejecuten las instrucciones de la lista	Ambos*
SETC<número>	Pone a la tortuga en curso del color indicado por <número>	Comando
COLOR	Devuelve el número que indica el color de la tortuga en curso	Operador

<nu/lis>: número de tortuga o lista de números de tortugas.
<lista ins>: lista de instrucciones.
* ASK se comporta como operador o comando dependiendo de la lista de instrucciones que lo acompañen.

TORTUGAS DE COLORES

Al igual que el fondo y las tizas, también las tortugas pueden cambiar de color.



Al igual que el fondo de la pantalla y las tizas de dibujo, también la tortuga puede adoptar varios colores. El comando al efecto es SETC, seguido por el número correspondiente al color seleccionado.

Las instrucciones al efecto son SETC y COLOR. SETC, seguido por un número, otorgará a la tortuga o tortugas en curso el color especificado por dicho número.
A su vez, el operador COLOR devuelve el número o código del color adoptado por las tortugas actualmente en uso.
Ambas órdenes pueden asociarse tanto con EACH como con ASK. Una observación a considerar es que el color de las tortugas es totalmente independiente de los colores elegidos para el fondo y las tizas.

¿UNA O VARIAS?

Todas las instrucciones presentadas para su ejecución con una sola tortuga pueden ser utilizadas con varias. Al respecto, sólo habrá que prever el uso de TELL, con lo que todas las tortugas en curso obedecerán las órdenes del programa. Cabe señalar también que por medio de ASK y EACH será posible diferenciar distintas órdenes para cada tortuga.

Los comandos del MP/M

Herramientas para el control de un entorno multiusuario

De cara al usuario, el MP/M se comporta de forma análoga al sistema operativo CP/M; por supuesto, con la evidente diferencia de que permite que varios usuarios se encuentren conectados al mismo ordenador, compartiendo los recursos de la CPU y la memoria y, en definitiva, mejorando el rendimiento de la instalación. Al igual que ocurría en el caso del CP/M, el sistema operativo MP/M ha de poner a disposición de los distintos usuarios toda una serie de herramientas que permitan gestionar el conjunto de datos almacenados en la memoria; algo sumamente importante en un sistema que permite el acceso simultáneo de hasta 16 usuarios. Además, por su propia naturaleza de sistema operativo multiusuario, el MP/M ha de brindar también los medios adecuados para el control de los periféricos conectados al ordenador (unidades de disco, impresoras, ...). Estas herramientas deben proteger a los distintos periféricos del acceso simultáneo de varios usuarios y, al tiempo, facilitar el acceso concurrente o secuencial a los mismos.
En definitiva, el sistema operativo ha de garantizar la correcta y eficaz explotación de los recursos del equipo.

LOS COMANDOS DEL MP/M

A raíz de lo señalado en los párrafos anteriores, es posible establecer una primera clasificación de los comandos que el MP/M pone a disposición de los usuarios del sistema.

- *Comandos destinados a la gestión interna y externa de los ficheros.*

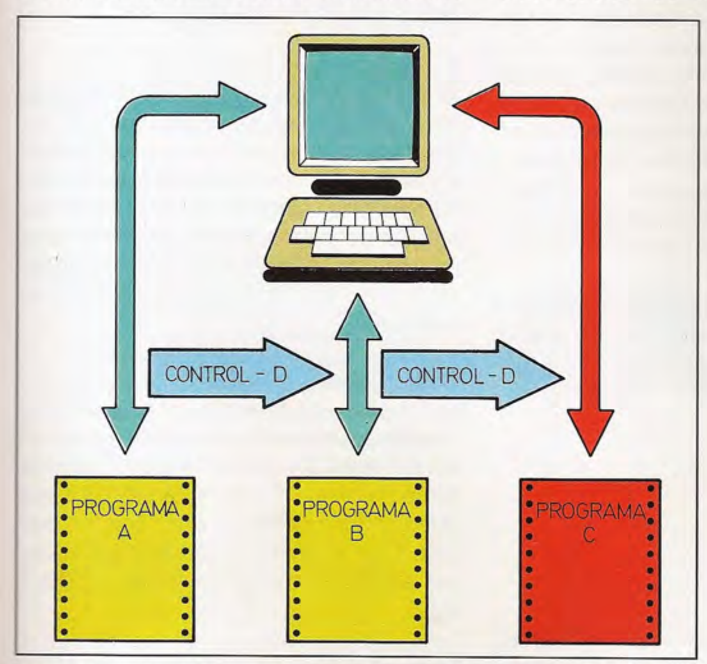
Dentro de este grupo, y a modo de ejemplo, cabe mencionar al editor (comando ED) o al comando PIP, el cual permite la copia de ficheros.

- *Comandos destinados al control de los periféricos.*

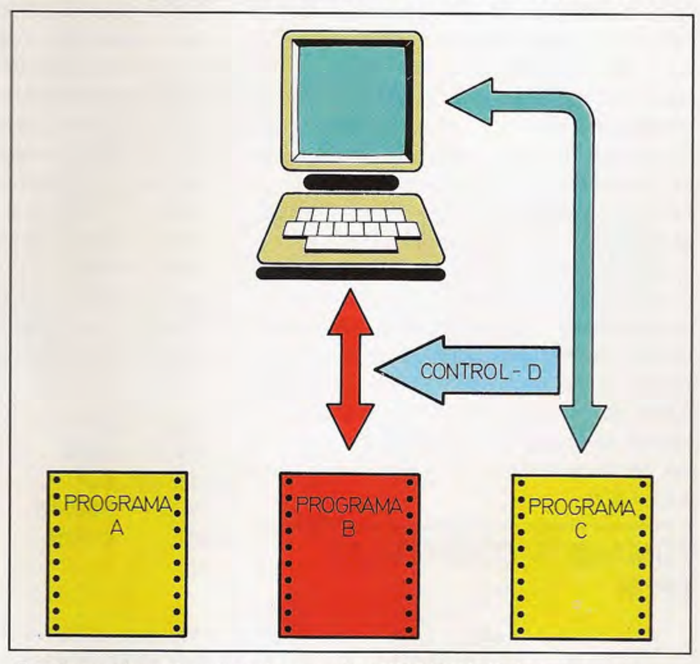
En esta categoría caben comandos especializados en la gestión y el control de los dispositivos auxiliares que apoyan la actividad del ordenador; por ejemplo, CONSOLE y SPOOL.

- *Comandos para el control de los programas de aplicación que se encuentren activos en el ordenador.*

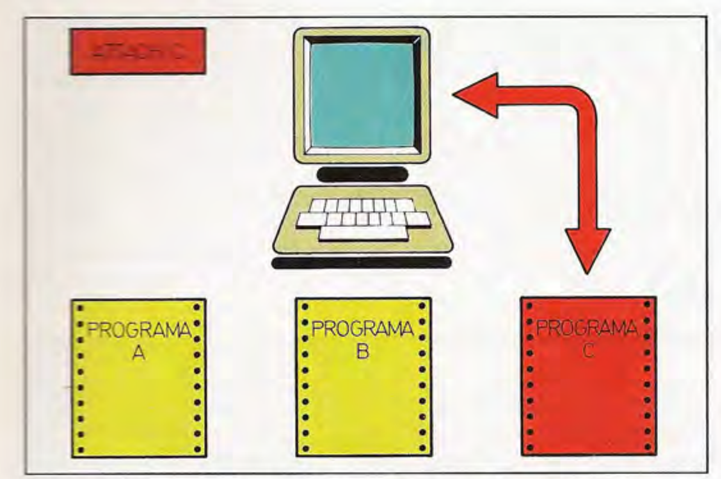
ATTACH, ABORT o SCHED son coman-



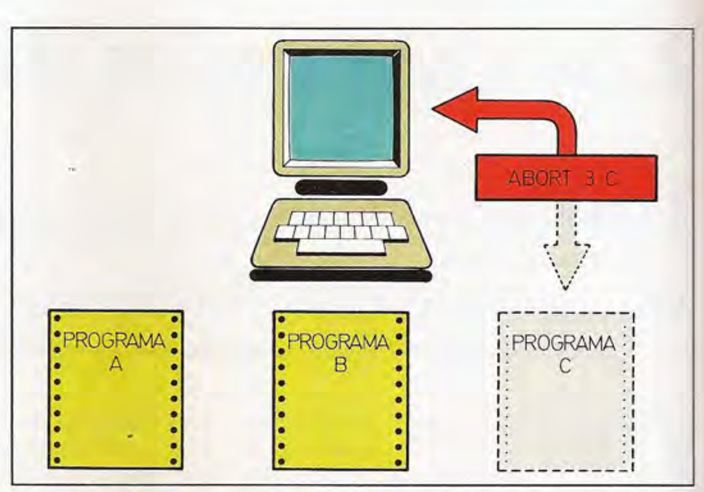
Cada ejecución del comando CONTROL D libera a la consola del control de un programa. Una vez liberada, ésta puede pasar a controlar la ejecución de cualquier otro de los programas que le resulten accesibles.



La ejecución reiterada del comando CONTROL D da lugar a un proceso reversible. El ordenador irá devolviendo a la consola el control de los programas liberados por efecto de la ejecución previa del referido comando.



La misión del comando ATTACH es devolver al terminal que lo activa el control de un proceso o programa desligado del mismo por efecto de la orden CONTROL D. En este caso, la recuperación del control es selectiva, ya que permite elegir directamente cuál de los programas debe quedar de nuevo bajo control.



El MP/M dispone de un comando adecuado para forzar el abandono de la ejecución de un programa antes de que éste concluya por sí mismo: ABORT. En su argumento hay que incluir el número del terminal desde el que se activó el programa (consola 3 en el ejemplo), así como el nombre del programa afectado.

dos integrados en esta tercera categoría. En esta clasificación están integrados varios comandos que el MP/M comparte con su equivalente monousuario, el CP/M. Este es un hecho previsible dada la analogía que existe entre ambos sistemas operativos. De hecho, la mayor parte de los comandos incluidos en el primer grupo de la clasificación precedente, son comandos del CP/M. Desde luego, en algunos casos, presentan una sintaxis que difiere ligeramente. No hay que perder de vista que el ámbito de actuación del MP/M obliga a precisar detalles adicionales. Por ejemplo, debido a la multiplicidad de usuarios, pueden existir ficheros con el mismo nombre en usuarios diferentes, circunstancia que debe ser considerada en la sintaxis de los comandos MP/M.

Los comandos MP/M pueden clasificarse, al igual que en el caso del CP/M, en comandos *permanentes* y *transitorios.* Los comandos del primer tipo son aquellos que se cargan en la memoria del ordenador al mismo tiempo que el sistema operativo. A su vez, los transitorios residen fuera de la memoria central; su traslado desde el disco a la memoria interna sólo se produce cuando el usuario invoca su ejecución.

COMANDOS COMPATIBLES CP/M

En efecto, la mayor parte de los comandos del sistema operativo CP/M son utilizables en los equipos gobernados por el MP/M. Los comandos compartidos de tipo permanente son introducidos en el ordenador en el momento de la carga, mientras que los transitorios han de aparecer como ficheros del tipo PRL (Program Relogeable). Estos últimos son utilizables por todos los usuarios del sistema. Al ser invocados, se cargarán en uno de los segmentos de memoria reservados al usuario de quien procede la solicitud.

El sistema operativo MP/M incorpora utilidades para la transformación de ficheros que contengan comandos transitorios y no sean de tipo PRL. En efecto, pueden existir comandos con un formato inadecuado, o que presenten direccionamientos de memoria absolutos y no relativos a los distintos segmentos en los que distribuye la memoria el sistema operativo MP/M; ficheros cuya adecuación supone transformarlos a tipo PRL.

Los comandos CP/M compatibles con el sistema operativo MP/M son los que se relacionan a continuación:

DIR	ED
ERA	LOAD
REN	DDT
TYPE	SUBMIT
STAT	DUMP
PIP	

Hay que tener en cuenta que el formato que adopten en el MP/M no tiene por qué coincidir. Ello dependerá básicamente de la función que realicen y de su interacción con los restantes usuarios del sistema.

COMANDOS PROPIOS DEL MP/M

Dentro de los comandos propios del MP/M o que tienen una significación particular del mismo, cabe establecer dos categorías: caracteres de control y comandos o utilidades suplementarios.

- *Caracteres de control*

El primer grupo engloba a las órdenes cuya introducción se realiza con una acción simultánea sobre la tecla de control (CTRL) y una tecla alfabética. Su misión primordial consiste en el control de los programas y procesos en curso de tratamiento por el ordenador. A continuación se definen los caracteres de control más relevantes.

CONTROL-S

La función de esta orden es interrumpir el flujo de presentación de datos en el terminal del operador que lo solicite. Suspende temporalmente la presentación para que puedan examinarse los datos con comodidad. Para su introducción hay que accionar simultáneamente las teclas <CTRL> y S.

CONTROL-Q

Indica a la consola o terminal que puede reasumir el proceso de edición de datos por pantalla, interrumpido por efecto de un co-

mando CONTROL-S. En el caso de pulsar esta combinación inadvertidamente (<CTRL> y Q) no se produce ningún efecto pernicioso en el funcionamiento del ordenador, ya que sólo es operativo cuando aparece interrumpido el proceso de edición.

CONTROL-D
El comando CONTROL-D hace posible una de las facetas características del MP/M la multiprogramación desde una misma consola. Desde un mismo terminal puede ordenarse la ejecución de más de un programa. Una vez que el primer programa ha recibido los datos necesarios para su operación, y se ha iniciado el proceso de cálculo, es posible liberar la consola del mismo y ordenar la ejecución de un segundo programa. Este proceso puede repetirse sucesivamente hasta que todos los segmentos de memoria se encuentran asignados.
Resulta evidente que esta facultad es útil cuando los programas en ejecución exigen constantemente recursos de CPU y no tienen interacción con el usuario más que en la introducción de datos y, tal vez, en la notificación de resultados.
El proceso es reversible, de cara precisamente a permitir la interacción con el usuario, al introducir la orden CONTROL-D. El ordenador devuelve a la consola el control de los programas, si bien, el proceso de devolución tiene lugar en el mismo orden en el que los programas fueron liberados de dicho control.

CONTROL-C
Permite interrumpir la ejecución de un programa y liberar la zona de memoria que éste tiene reservada. Este comando puede emplearse también tras la activación de una orden de interrupción del tipo CONTROL-S.
Es importante señalar que aunque los caracteres de control son herramientas estandarizadas del sistema operativo MP/M, pueden existir programas que anulen su efecto y den vigencia a otras combinaciones específicas de la tecla CONTROL. Un ejemplo de ello lo constituyen algunos programas para el tratamiento de texto, como es el caso del WORDSTAR.

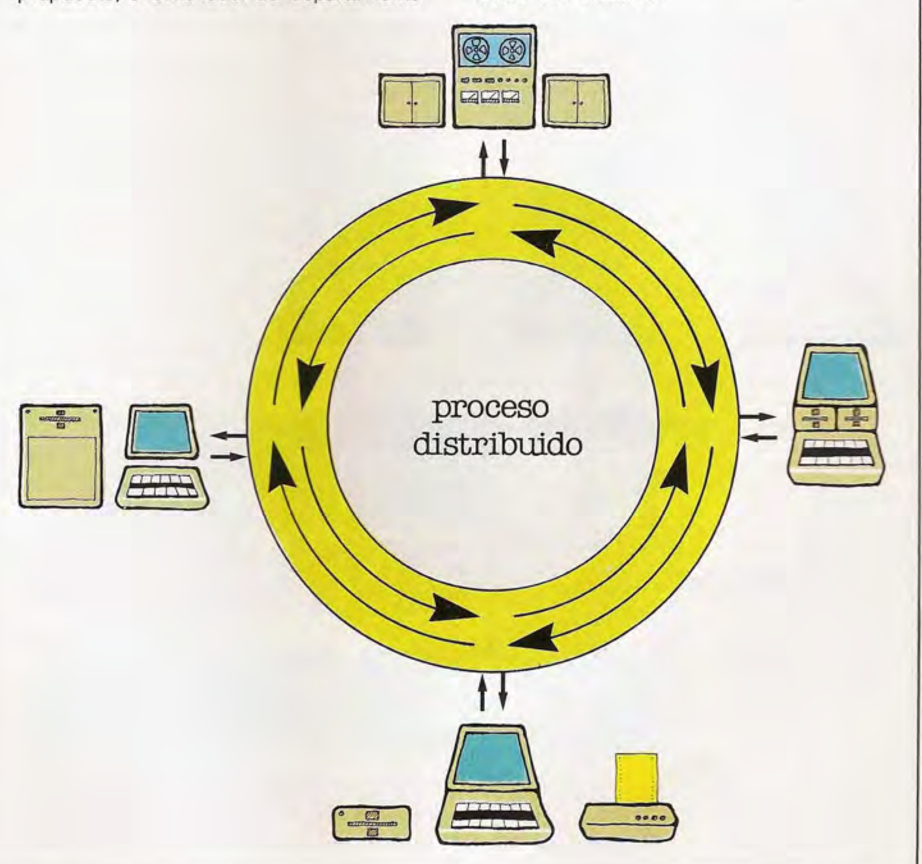
- *Comandos suplementarios y utilidades del MP/M*

Los más característicos son los que se relacionan a continuación, acompañados por una breve descripción de su funcionamiento.

DIR (SYS)
Su cometido es visualizar la lista de fiche-

Proceso distribuido

En la actualidad y debido al coste cada vez inferior de los microprocesadores y elementos de memoria, ha sido posible el nacimiento de una amplia variedad de microordenadores de distinta capacidad y potencia. A raíz de esta eclosión microinformática, cada día es mayor el número de usuarios que franquean el umbral de la automatización. Usuarios con unas necesidades específicas y cuya resolución exigen al equipo que obra a su disposición.
En este punto, y con un efecto dilatado del margen de actuación de equipos con recursos limitados, nace el proceso distribuido.
Partiendo de un grupo de ordenadores y, a través de un sistema de comunicación local o remoto, el proceso distribuido permite la puesta en común de recursos de los distintos equipos; de tal forma que cada uno de los ordenadores asociados tiene abierto el acceso a los recursos de los restantes equipos, tanto a nivel de programas como de ficheros de datos.
Un ejemplo elocuente cabe encontrarlo en el seno de una gran corporación comercial. En ella, el ordenador que controla los resultados de los procesos de fabricación, está interconectado con el ordenador del departamento de ventas. Tal comunicación garantiza el acceso de este último a los datos procesados en el primero, lo que permite una constante evaluación de las posibilidades comerciales de la compañía.
El proceso distribuido aporta sustanciales ventajas, no siendo la menor de ellas el hecho de que cada usuario obtiene el máximo rendimiento de la máquina que está a su disposición, sin emplear más recursos que los estrictamente necesarios para su labor. Así, en el caso del ejemplo propuesto, el ordenador del departamento de ventas no necesita tener duplicada la información contenida en el de producción, con lo que su capacidad puede ser inferior y sin que ello degrade la respuesta que obtiene el usuario.



COMANDOS DEL MP/M

Comando	Significado
DIR[SYS]	Visualiza un directorio incluyendo ficheros de sistema.
CONSOLE	Indica número de consola.
ERAQ	Borrado de ficheros.
DSKRESET	Permite al usuario cambiar disquetes.
GENHEX	Genera un fichero hexadecimal de un fichero COM de CP/M.
PRLCOM	Genera un fichero COM de CPM a partir de un fichero PRL.
GENMOD	Crea un fichero PRL de un fichero hexadecimal.
SPOOL	Envía un fichero a las colas de impresión.
STOPSPRL	Detiene la salida de impresión.
TOD	Indica fecha y hora.
SCHED	Prepara una tarea para ejecutar a una hora fijada.
ABORT	Detiene la ejecución de un programa.
ATTACH	Devuelve el control de una tarea.
PRINTER	Selecciona la impresora a emplear.
SET	Indica estado del disco e introduce palabras de paso.
SHOW	Muestra estado del disco.
MPMSTAT	Estado del sistema operativo.
SDIR	Visualización del directorio.

ros de un directorio, incluyendo a los ficheros del sistema o ficheros comunes a todos los usuarios del equipo. Asimismo, permite la consulta del directorio de ficheros propio de cada usuario específico.
El formato de esta orden es variable; depende de la versión del sistema operativo MP/M que se utilice.

CONSOLE
Permite consultar el número de la consola o terminal que está utilizando un determinado usuario. Su presencia es necesaria, habida cuenta que el MP/M soporta hasta 16 consolas y este número es necesario en la formulación de ciertos comandos.

ATTACH
Este comando tiene encomendada la misión de devolver a la consola el control de un programa o proceso, después de que este haya sido desligado de la misma por efecto del comando CONTROL-D. Permite seleccionar cuál de los programas activos en un momento determinado debe quedar de nuevo bajo control. A la luz de lo comentado, se observa que ATTACH es un comando más selectivo que CONTROL-D en su faceta de devolución de control. Su formato es:

A>*ATTACH Nombre del programa*

ABORT
Fuerza el abandono de la ejecución de un programa antes de que éste concluya por sí mismo. En el caso de que el programa haya sido llamado desde una consola diferente, habrá que incluir en el formato de la orden el número de la consola o terminal en cuestión. Hay que tener en cuenta que la ejecución del programa puede haberse activado desde otro terminal del sistema. El formato de la orden es:

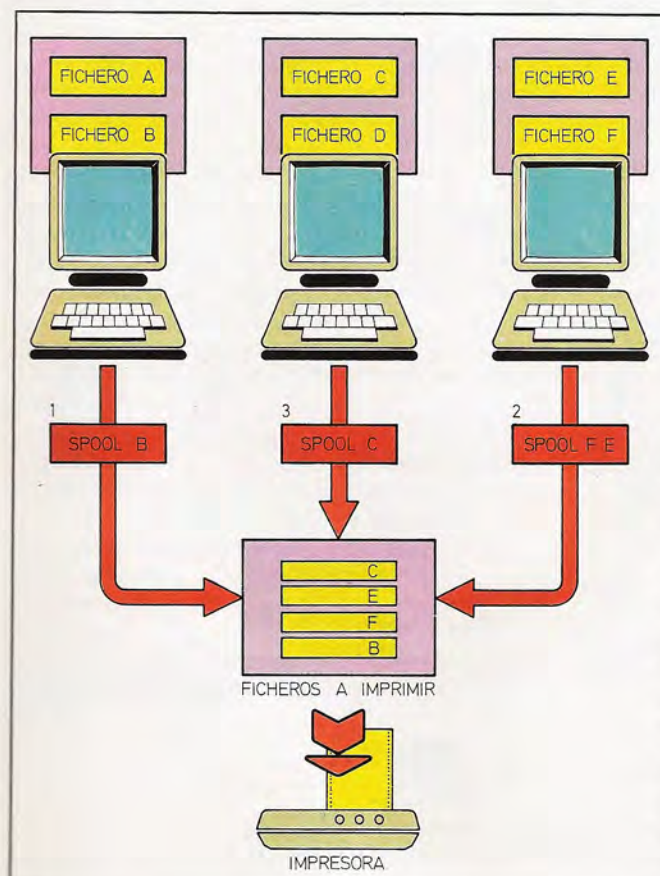
A>*ABORT n Nombre del programa*

El número "n" es opcional, refleja el número de consola desde la que fue llamado el programa.

SPOOL
La función de este comando es permitir el uso concurrente, por parte de varios usuarios, de una única impresora. Para ello, el ordenador envía la información de salida a un fichero en disco y crea unas colas para salida a través de impresora a medida que van llegando nuevos ficheros para imprimir. La referida información de salida es canalizada hacia la impresora de forma secuencial, en el mismo orden en el que fue incorporándose a la cola de impresión. El formato del comando SPOOL es el siguiente:

A>*SPOOL Nombre de fichero Salida*

La orden admite la expresión de varios ficheros de salida; la única limitación en este punto la impone la longitud de la línea de órdenes.



En el ámbito del MP/M, los distintos usuarios pueden compartir una misma impresora asociada al sistema. La herramienta que permite el uso concurrente de la misma la aporta el comando SPOOL; éste permite trasladar los archivos a imprimir a una "cola" de espera. La impresora irá dando curso a los distintos archivos de acuerdo al orden en el que hayan ingresado en la cola de salida.

Visicalc (1)

Las facultades de una hoja electrónica versátil y popular

El paquete de aplicación VISICALC es, sin lugar a dudas, uno de los más populares y presentes en el mercado del software de aplicación. Su autoría se debe a la firma americana Visi Corp.

El éxito del paquete VISICALC se debe, entre otros condicionantes, a que fue la primera hoja electrónica que llegó al mercado. Otro factor también importante estriba en la profusión de versiones que existen, lo que permite su empleo en un gran número de ordenadores personales, desde los distintos modelos de Apple, hasta el doméstico Atari, pasando por el IBM-PC y todos sus compatibles.



Para utilizar con eficacia las posibilidades del VISICALC es preciso que el usuario se familiarice con el teclado de su ordenador. Las teclas que aparecen en la figura son las de uso más frecuente.

PRESENTACION DEL VISICALC

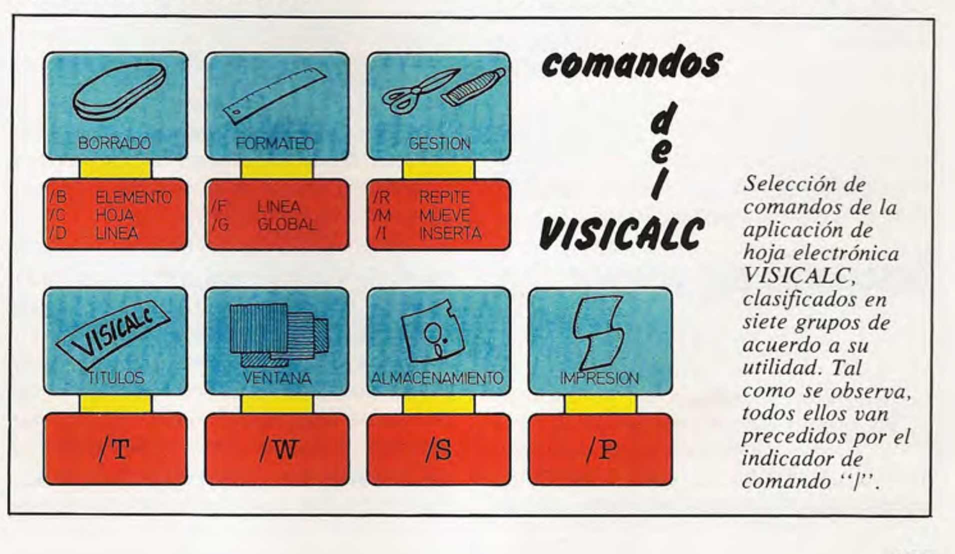
Inicialmente, la hoja electrónica VISICALC se diseñó para la resolución de problemas financieros y de gestión por medio de ordenadores personales. Se intentaba lograr, de esta forma, que los usuarios finales pudieran utilizar el ordenador para resolver sus problemas concretos, sin necesidad de desarrollar programas específicos para cada uno de ellos.

La evolución de esta hoja electrónica ha sido constante, hasta el punto de que en la actualidad existen versiones de distinta potencia, entre las que cabe destacar como más modernas el Adv. VISICALC y el VISICALC IV. Ello permite al usuario elegir entre ellas la variante más apropiada a sus necesidades y capacidad económica. Además de poder decidir entre las versiones existentes dentro de la línea oficial del producto, existen muchas otras empresas de software que han desarrollado productos auxiliares o complementarios.

Al principio se ha señalado que el VISICALC está disponible para varios ordenadores personales, de distintos fabricantes. En consecuencia, es obvio que existen dialectos preparados para trabajar bajo distintos sistemas operativos como, por ejemplo, el CP/M, MS/DOS, Apple DOS y, en general, cualquiera de los sistemas operativos más extendidos.

CARACTERISTICAS TECNICAS DEL VISICALC

La aplicación utiliza la pantalla del ordenador personal como ventana de una matriz/hoja, en la que el usuario puede realizar



Selección de comandos de la aplicación de hoja electrónica VISICALC, clasificados en siete grupos de acuerdo a su utilidad. Tal como se observa, todos ellos van precedidos por el indicador de comando "/".

sus "anotaciones". El tamaño físico de la matriz se eleva a un máximo de 254 filas y 63 columnas.
Las filas se numeran, como es tradicional, mediante números consecutivos 1, 2, 3, ..., 252, 253, 254; sin embargo, las columnas se identificarán mediante letras A, B, C, ..., BI, BJ, BK.
Las "medidas" de la hoja son lo suficientemente grandes como para que el tamaño de la pantalla no permita representar toda la información de la hoja completa de una sola vez. De ahí que la pantalla se utilice a modo de ventana; ventana que puede desplazarse sobre la hoja electrónica para examinar y/o modificar la zona adecuada. Se puede realizar un SCROLL de la ventana o pantalla a través de la hoja total, si bien, cuando lo que se desea es desplazarse rápidamente a través de la misma, resulta mucho más apropiado utilizar el comando de salto (/ >). Este último permite ir directamente a la casilla cuya letra de columna y número de fila se especifican.
Además de las teclas especiales del terminal en el que se esté trabajando, existen dos tipos de operadores aportados por el VISICALC: *funciones* y *comandos*. Las funciones se pueden utilizar para las fórmulas de cálculo, con las que se definirán los elementos variables de la matriz, mientras que los comandos resultan adecuados para gestionar la hoja electrónica.
Para poder utilizar eficazmente el VISICALC, es imprescindible que el usuario conozca el teclado de su ordenador. Este será el intermediario entre sus deseos y la hoja. Las teclas que se utilizarán con mayor frecuencia son las que aparecen en la tabla adjunta.
Además de estas teclas especiales, en una sesión de trabajo con VISICALC será necesario utilizar las teclas alfabéticas para introducir los literales y las teclas numéricas para introducir los datos.
Para completar el análisis de las características técnicas del VISICALC, a continuación se detallan los comandos y funciones más relevantes que incorpora.

COMANDOS DEL VISICALC

Todos los comandos que el VISICALC pone a disposición del usuario se identifican por medio de una letra precedida por el símbolo "/". De esta forma, el programa sabrá diferenciar entre una letra representativa de un comando y una letra que identifica a una columna de la hoja electrónica. Para ejecutar un comando basta, pues, con pulsar la tecla "/" e introducir a continuación la tecla alfabética correspondiente. Los principales comandos del VISICALC son los que se relacionan en los siguientes apartados.

1. BORRADO DE UN ELEMENTO

Se identifica mediante la letra B y sirve para eliminar el contenido del elemento en que se encuentra el cursor. Suele utilizarse para corregir errores.

2. BORRADO DE UNA PAGINA

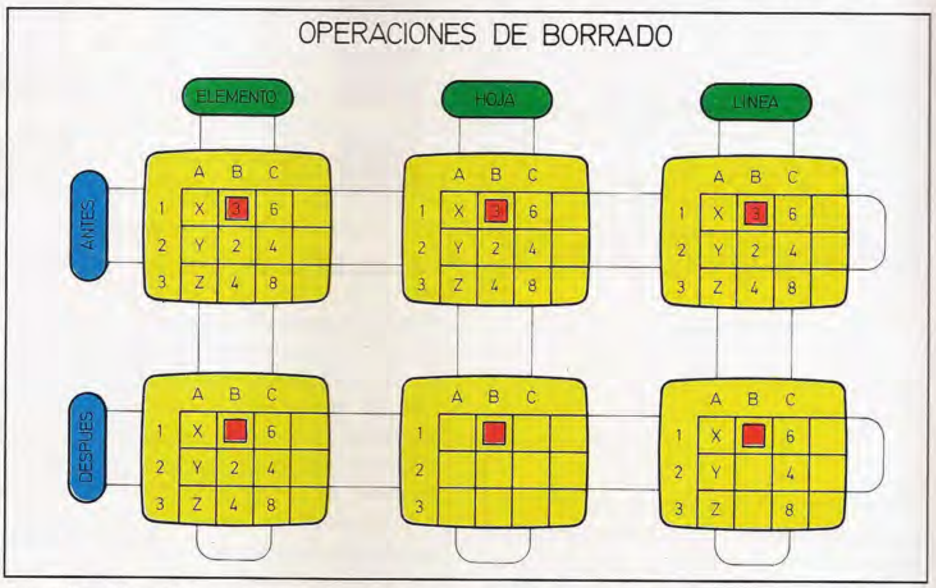
El comando adecuado se identifica por medio de la letra C. Su acción borra completamente el contenido de la hoja electrónica. Su empleo sólo es conveniente cuando ha concluido el trabajo en curso y se desea comenzar otro.
El campo de actuación de este comando se limita a la memoria principal del ordenador y, por lo tanto, no afecta a los datos que pudieran estar almacenados en disco.
No cabe duda que su uso puede resultar peligroso, ya que si se introduce por error, puede dar lugar al borrado de una hoja que interesa conservar. Para evitar este tipo de accidentes, el programa solicita la confirmación del comando, para lo que el usuario debe responder accionando la tecla "Y" (YES).

3. BORRADO DE UNA LINEA

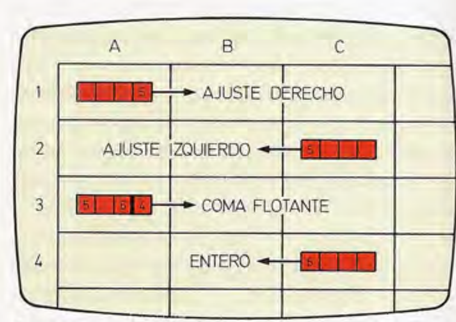
Corresponde a la letra D. La ejecución de este comando borra completamente el contenido de una fila o columna. Cuando el usuario lo invoca, y antes de producirse el efecto deseado, el VISICALC pregunta si la línea que se desea borrar es una fila o una columna. Acto seguido, según la respuesta del usuario, se eliminará la información contenida en la fila o columna en la que se encuentre situado el cursor.

4. FORMATEO DE UN ELEMENTO

Es activado por la letra F. Permite al usuario definir el modo en el que desea representar el contenido de un elemento. Según la letra pulsada a continuación de "/F", quedará seleccionado un formato

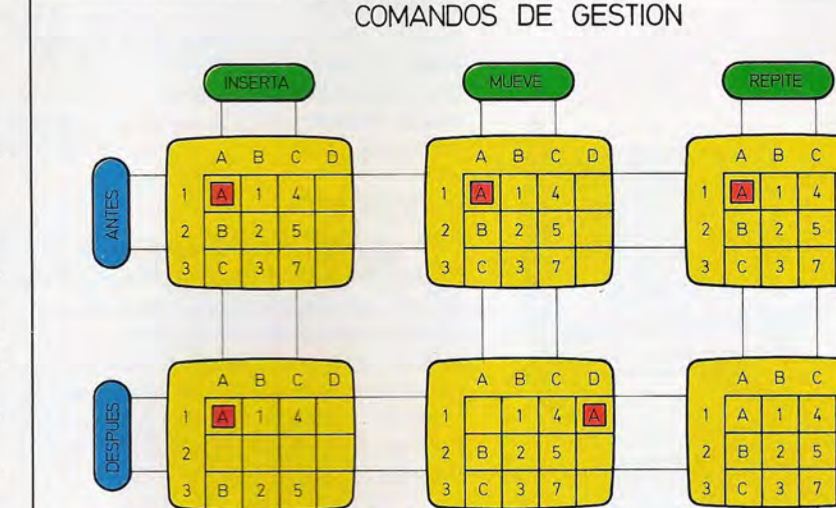


La aplicación ofrece tres comandos básicos para el borrado: de elementos, de líneas (fila o columna, según se especifique) o de la hoja completa. Su actuación queda reflejada en la figura adjunta.



Los comandos de formato permiten definir las características de representación de los distintos datos en las casillas o elementos de la hoja electrónica.

Insertar, mover y repetir o duplicar datos en distintos elementos, son operaciones básicas para el trabajo con la hoja electrónica. Las herramientas adecuadas al efecto cabe encontrarlas en el apartado de comandos de gestión del VISICALC.



específico; las posibilidades son las siguientes:

- D: Formato en coma flotante, esto es: representación de un número con tantos decimales como sea preciso.
- I: Formato entero y, por lo tanto, representación del número sin decimales.
- L: Formato ajustado al margen izquierdo del elemento o casilla.
- R: Formato ajustado al margen derecho del elemento.
- $: Formato con dos decimales.
- *: Formato para reemplazar el contenido del elemento por tantos asteriscos como su valor entero (útil para realizar gráficos).

5. FORMATEO DE UNA PAGINA

El comando al efecto se identifica mediante la letra G y sirve para especificar características de la hoja en la que se desea trabajar. Este comando se activa pulsando "/G" y una segunda letra que define la opción elegida:

- C: Define el ancho de una columna con un mínimo de tres caracteres.
- F: Define el formato de todos los elementos de una hoja. (Igual que el comando F, pero aplicado a todos los elementos y no a la celda específica en la que se encuentra el cursor).
- O: Permite especificar que el recálculo de elementos se realice por filas o por columnas según desee el usuario.
- R: Define la prioridad de recálculo; ésta puede ser automática o manual.

6. INSERCION

La utilidad del comando de inserción, representado por la letra I, es la de introducir una nueva fila entre dos ya existentes, o una nueva columna también entre dos columnas ya existentes en la hoja electrónica. Debe acompañarse de una segunda letra que puede ser:

- R: Insertar una fila bajo la que contiene al cursor.
- C: Insertar una columna tras la que contiene al cursor.

Elección de una hoja electrónica

La competencia dentro del mercado actual de hojas electrónicas, entre los distintos productos existentes, es tan grande que resulta enormemente complicado tomar una decisión de compra. Tras el éxito inicial de la aplicación VISICALC, la práctica totalidad de empresas de software se dedicaron a producir nuevos paquetes con mejoras respecto a su competencia.
La lucha por alcanzar el primer puesto dentro de la lista de productos más vendidos, ha acelerado el desarrollo de las aplicaciones horizontales de gestión y productividad hasta llegar a los modernos paquetes integrados. En éstos, la hoja electrónica se ve rodeada de otras aplicaciones complementarias que abren la posibilidad de producir gráficos, aportan facilidades de gestión y de edición...
Tan vertiginoso desarrollo siembra la incertidumbre en el usuario a la hora de decidir la compra de un determinado paquete de aplicación. Desde luego, la única forma de garantizar que el producto adquirido es el mejor, consiste en esperar unos años, y ver cuál de los programas termina ganando esta desenfrenada carrera. No obstante, esta fórmula resulta inviable en la mayoría de los casos. El usuario tiene que optar por una alternativa y ¡debe decidirlo ya! A continuación, se relacionan algunos criterios básicos que pueden ayudar a tomar la decisión final:

1. Compatibilidad con el equipo informático.

Si el usuario ya dispone de un ordenador personal, la primera criba la realizará atendiendo al subconjunto de hojas electrónicas que puedan funcionar en su equipo. Desde luego, si se molesta en analizar la oferta, encontrará varias aplicaciones de este tipo compatibles con su ordenador.

2. Fiabilidad del suministrador.

Otro punto importante a considerar es la reputación del vendedor del producto. En muchos casos es recomendable consultar con el suministrador del equipo que deba soportar la aplicación.

3. Facilidad de aprendizaje.

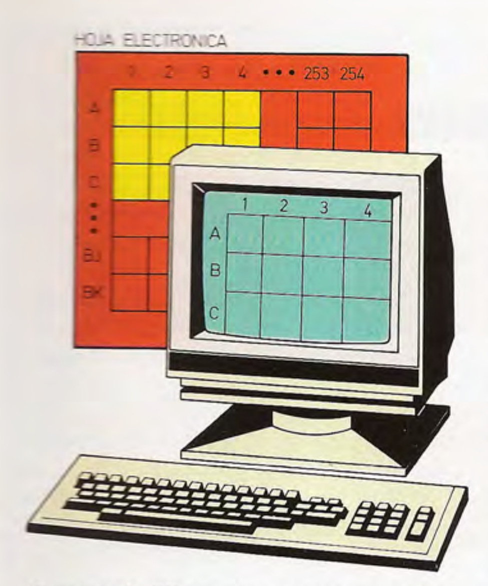
Para que el rendimiento de la aplicación sea óptimo, hay que evaluar la sencillez de su uso práctico y la disponibilidad de manuales o/y otros materiales didácticos.

4. Características técnicas del producto.

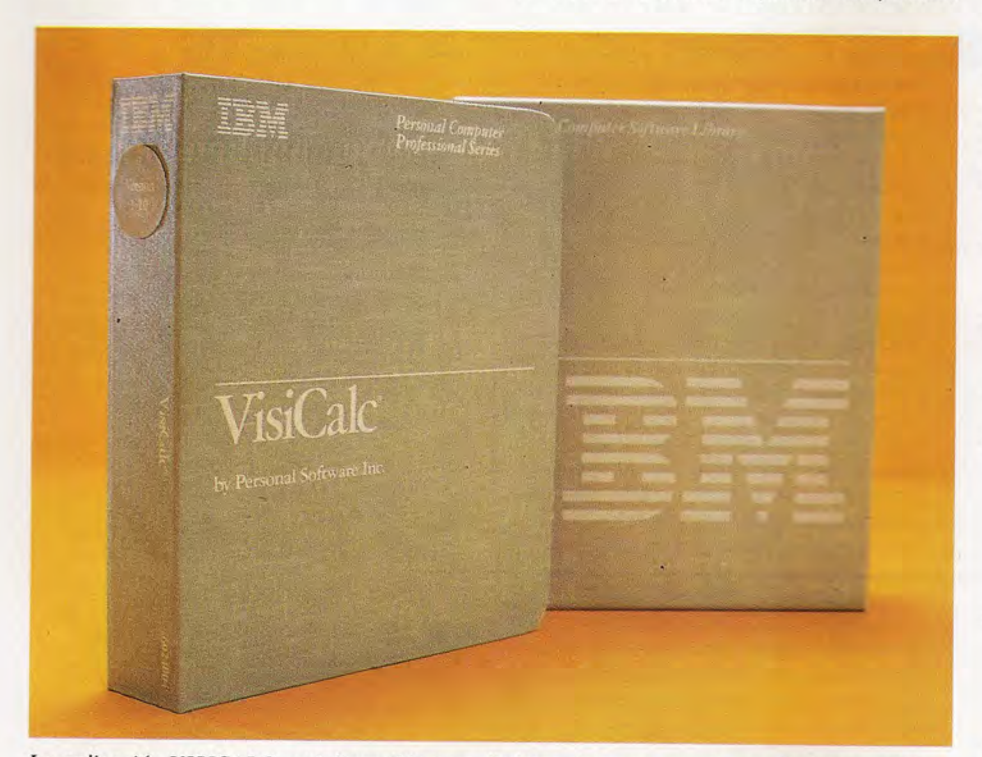
A los criterios relacionados hasta ahora, hay que añadir otro factor de total importancia: las propias características técnicas de la hoja electrónica. Entre ellas cabe destacar los siguientes puntos:

- Entrada de datos.
- Comandos disponibles.
- Tamaño de la "Hoja" (filas y columnas).
- Gestión de datos.
- Capacidad gráfica.
- Comodidad de uso.

Todos éstos son apartados cuyo análisis específico, para los principales productos del mercado, se realizará a lo largo de la obra.



El comando /W permite utilizar la pantalla a modo de ventana, capaz de desplazarse y visualizar los distintos fragmentos de la hoja electrónica almacenada en la memoria central del ordenador.



La aplicación VISICALC está disponible para distintos sistemas operativos y distintos modelos de ordenadores personales. El IBM-PC es uno de los equipos en los que su presencia es más frecuente.

7. MOVIMIENTO

La letra M se encarga de dar entrada a este nuevo comando. Se utiliza para mover la línea en la que se encuentra el cursor a una nueva posición. Para ello es imprescindible conocer, por un lado, si se desea mover una fila o una columna, y por otro lado las coordenadas del nuevo destino (el nuevo destino también se puede indicar llevando el cursor al mismo, si así se desea).

8. REPETICION

Este comando, representado por la letra R, permite repetir el elemento o la línea en que se encuentra el cursor, en otra posición dentro de la hoja electrónica. En el caso de que el elemento repetido sea una fórmula, el VISICALC preguntará al operador si desea modificar los parámetros en la referida fórmula en la nueva situación.

9. TITULOS

La letra T da paso a un nuevo comando del VISICALC cuyo objeto es reservar posiciones de la hoja cuyo único destino será almacenar títulos. Su puesta en práctica evitará errores: si una vez hecha la reserva se pretende introducir un dato o una fórmula, con un nuevo comando o directamente, en alguna de dichas posiciones, el VISICALC avisará del error. Dado que los títulos pueden colocarse en disposición vertical y horizontal, el comando exige una segunda letra que indique el sentido del título. Las letras destinadas al efecto son:

H: Título horizontal.
V: Título vertical.

10. DEFINICION DE VENTANAS

Este comando, identificado mediante la letra W, permite definir las ventanas dentro de la hoja, para ajustar la zona visible al tamaño de la pantalla.
Existen cuatro posibilidades para la definición de ventanas cuya selección corre a cargo de las siguientes letras:

H: Ventanas horizontales.
V: Ventanas verticales.
S: Ventanas con paso sincronizado.
U: Ventanas con paso desincronizado.

11. GESTION DEL ALMACENAMIENTO

El comando al efecto se identifica mediante la letra S y permite traspasar información de la hoja en curso a un soporte de almacenamiento externo. Para indicar el tipo de operación deseado, hay que elegir entre alguna de las siguientes opciones:

L: Cargar en memoria, una hoja residente en disco.
S: Grabar en disco la hoja electrónica que se encuentra en la memoria central del ordenador.
D: Borrar un fichero en disco con información de una hoja electrónica.
I: Inicializar (preparar) un disco destinado a almacenar hojas electrónicas.
Q: Terminar la sesión de trabajo con el VISICALC sin grabar en disco la información. Dado que con esta opción se puede perder información útil, el programa pide al usuario que la confirme mediante la letra "Y" (YES).
#: Gestionar de forma conjunta varias hojas electrónicas, con objeto de incorporar en la hoja actual otra hoja electrónica o parte de ella.

12. IMPRESION

A través del comando de impresión, letra P, es posible definir el tipo de impresora conectada al equipo y ordenar la impresión en papel de una hoja electrónica.
El próximo capítulo se ocupará del estudio de las funciones que puede manejar el VISICALC. Ello constituirá la etapa previa a una sesión práctica de trabajo con esta popular hoja electrónica.

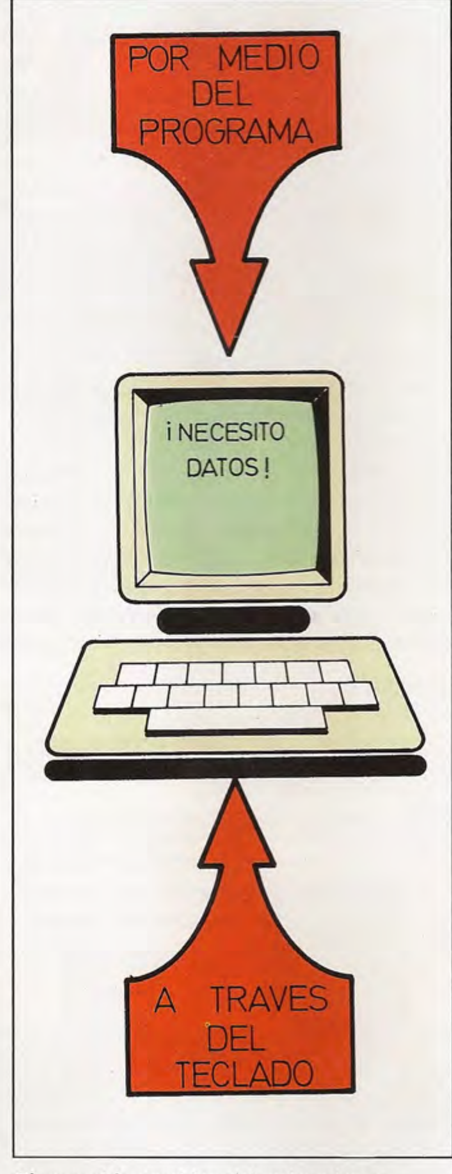
Aportando datos a la máquina

Los comandos READ, DATA y RESTORE

Cualquier programa, sea cual fuere su naturaleza (de gestión, educativo o, sencillamente, lúdico) necesita manipular una cierta cantidad de datos, que deben ser entregados a la máquina para que los elabore. Estos pueden incluirse dentro del propio programa, o incluso puede suministrarlos el usuario, en un determinado momento, a través del teclado.

DIFERENCIACION DE LOS DATOS

En términos generales cabe hablar de dos tipos de datos o, más exactamente, de dos formas de aportarlos al programa: como valores constantes o a modo de variables.
Aún cabe establecer entre los datos otra división, quizá no tan clara y definida como la anterior, pero no por ello menos válida. Esta división nace de su propia utilidad: ciertos datos van a mantener valores fijos y determinados en las sucesivas ejecuciones del programa; por el contrario, otros verán alterado su valor, ya que el programa los modificará para ofrecer al usuario el resultado de la ejecución.
Un ejemplo clarificará este extremo. Si se quiere obtener al cabo de un año el total de gastos mensuales de una familia, habrá que utilizar al menos dos variables con distinta función. Por un lado, el programador debe tener presente que el año consta de doce meses y, por lo tanto, habrá que considerar doce entradas de datos, una para cada mes del año. Por otro lado, hay que contar con otra variable que irá acumulando las sucesivas cantidades mensuales.
Un programa capaz de realizar estos cálculos es, por ejemplo, el que sigue:



El cometido genérico del ordenador es manipular datos de acuerdo a las indicaciones aportadas por el programa. Los datos pueden suministrarse a través de dos vías principales: el teclado y formando parte del propio programa.

```
10 LET SUMA=0
20 FOR M=1 TO 12
30 INPUT A
40 LET SUMA=SUMA+A
50 NEXT M
60 PRINT "GASTO ANUAL="; SUMA
70 END
■
```

Su ejecución interrogará al usuario acerca de los sucesivos gastos mensuales, hasta completar la introducción de datos mensuales; tras ello, mostrará el valor del gasto anual.

```
RUN
?35650
?42500
...
?15250
GASTO ANUAL=525345
```

El programa propuesto comienza por inicializar a cero la variable SUMA, para acumular en ella los doce valores, ingresados a través del teclado y asignados a la variable A. Esta forma parte de un bucle que se ejecutará doce veces.
Las dos variables en juego, SUMA y A, adoptarán valores muy diferentes en cada ejecución del programa: los gastos de un mes serán, por lo general, muy distintos a los de otro. Sin embargo, la variable de control del bucle FOR/NEXT (M), siempre adoptará los valores comprendidos entre uno y doce.
Desde luego que éste es un ejemplo trivial; en todo caso, existen muchas variables dentro de un programa que tienen el carácter fijo de la variable M. Por ejemplo: los nombres de los meses, el nombre y la fecha de nacimiento de los miembros de una familia o de los trabajadores de una

empresa, los resultados de los partidos de fútbol del año anterior... A este tipo de variables de carácter fijo, se les podría asignar, como es lógico, sus valores mediante instrucciones INPUT. Valores que se introducirán a través del teclado cada vez que se ejecute el programa. No obstante, y dado que estos valores son siempre los mismos, es obvio pensar que se ahorraría mucho tiempo y molestias, si su asignación corriera a cargo del propio programa.

Una forma elemental de almacenar los referidos datos por medio del programa la aporta la instrucción LET, adecuada para asignar directamente los valores deseados a las variables. Así, por ejemplo, se podrían almacenar los nombres de los días de la semana de la siguiente forma:

```
10 LET D1$="LUNES"
20 LET D2$="MARTES"
30 LET D3$="MIERCOLES"
40 LET D4$="JUEVES"
50 LET D5$="VIERNES"
60 LET D6$="SABADO"
70 LET D7$="DOMINGO"
■
```

Ello hará que el programa los tenga a su disposición en cualquier momento, obviando la necesidad de introducirlos cada vez que se desee ejecutarlo. Si el número de variables fuese reducido, ésta sería una opción viable y bastante eficaz, pero ¿qué ocurriría si el número de variables de este tipo fuese elevado?

LEYENDO LOS DATOS

Naturalmente, si hubiera que almacenar, por ejemplo, los resultados de los partidos de fútbol de la temporada pasada a base de instrucciones LET o INPUT, sería una tarea larga y tediosa. Afortunadamente, el BASIC cuenta con otras instrucciones que resuelven este problema con mayor comodidad.

Las referidas instrucciones son READ y DATA. Ambas deben coexistir obligatoriamente dentro de un programa, esto es: si se utiliza una instrucción READ, debe existir forzosamente al menos una instrucción DATA. La razón se comprenderá fácilmente una vez que se conozcan las funciones que realizan cada una de ellas y que, en esencia, se resumen en lo siguiente: las instrucciones DATA contienen los valores fijos que se desea asignar posteriormente a las variables, mientras que las instrucciones READ se encargan de leer el argumento de la instrucción DATA y asignar los valores de su argumento a las variables especificadas en la instrucción READ.

FORMATOS DE READ Y DATA

Como toda instrucción BASIC, tanto READ como DATA deben ajustarse a un formato sintáctico que es preciso respetar escrupulosamente. De lo contrario, la instrucción no será aceptada por el intérprete BASIC y la máquina generará un mensaje de "error sintáctico".

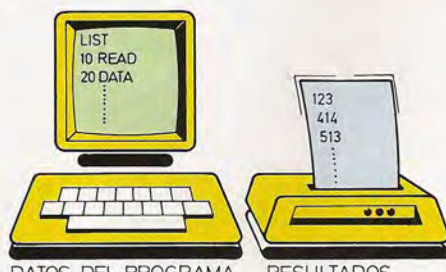
El formato de la instrucción READ es el siguiente: (Núm. línea) READ <var.>], <var.>, ..., <var.>]

La palabra clave READ debe ir precedida por el siempre obligatorio número de línea, propio de las instrucciones formuladas en modo indirecto (dentro de un programa). Su argumento incluirá a una serie de variables, separadas por comas, cuyo número es opcional; aunque siempre debe estar presente al menos una variable. Por ejemplo:

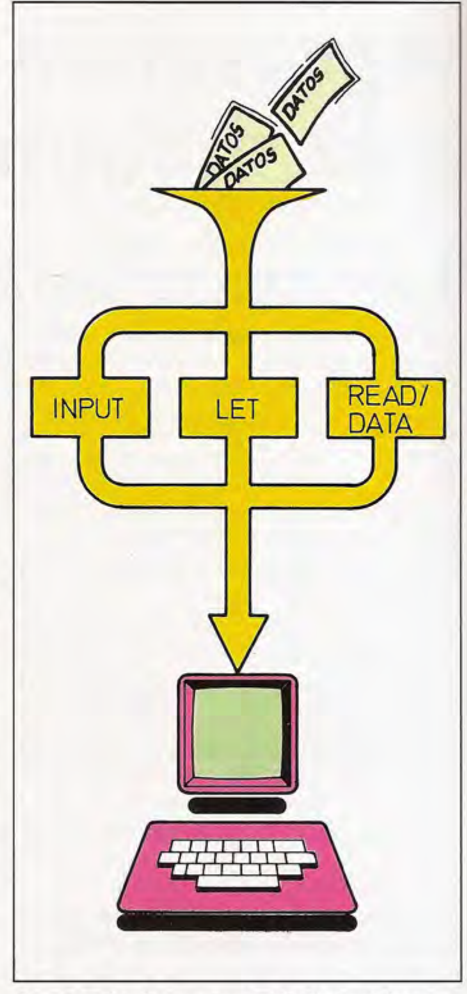
```
10 READ A
20 READ A, B, C$
```

A su vez, el formato de una instrucción DATA es el que sigue:

(Núm. línea) DATA <cte.> [, <cte.>, ..., <cte.>]



En términos generales, los datos pueden incluirse dentro de los programas adoptando la forma de variables o de valores constantes. Su tratamiento dará lugar a la generación del resultado o resultados del proceso.



Son tres las herramientas fundamentales que proporciona el BASIC para suministrar datos a la máquina; éstas coinciden con las instrucciones INPUT, LET y con la asociación READ/DATA.

Este formato es muy similar al anterior. Ahora, la palabra clave, DATA, irá seguida por al menos un dato; éste puede ser una constante de tipo numérico o alfanumérico (cadena de caracteres). Normalmente serán varios los datos incluidos en el argumento y que aparecerán separados por comas:

```
10 DATA 1,2,5,20,3
20 DATA LUNES
```

Las instrucciones DATA son leídas por las instrucciones READ, de acuerdo al orden que establecen sus números de línea aunque éstos no sean correlativos. En muchos dialectos BASIC los datos alfanuméricos pueden escribirse sin encerrarlos entre comillas; por supuesto, excepto en el caso de que el literal contenga alguna coma (,), con objeto de que la máquina no



La actividad del ordenador supone la manipulación de datos, la ejecución de cálculos y, en definitiva, el tratamiento de los mismos de acuerdo a la secuencia de órdenes denominada programa.

la confunda con las comas utilizadas para separar a los distintos datos. Esta observación es extensiva a las cadenas de caracteres que incluyan el signo dos puntos (:), para no confundirlo con el separador de dos instrucciones en la misma línea, o espacios en blanco significativos detrás o delante de la constante. Por ejemplo:

```
DATA MARTIN, "SANCHEZ," ,ENRIQUE
DATA "HORA:", "MINUTOS "
```

Las instrucciones DATA incluirán valores de tipo numérico o literal. No se admiten expresiones numéricas, ya sea con operaciones matemáticas o lógicas entre los datos.

Así, por ejemplo, no serán aceptadas las siguientes instrucciones:

```
10 DATA 3/4
20 DATA A AND B,C OR B
```

En ambos ejemplos, los valores aportados en las sentencias DATA podrían ser leídos como cadenas de caracteres, pero no resultarán válidos si su misión es aportar el valor resultante de las expresiones aritméticas o lógicas.

READ-DATA: UNA PAREJA INDISOLUBLE

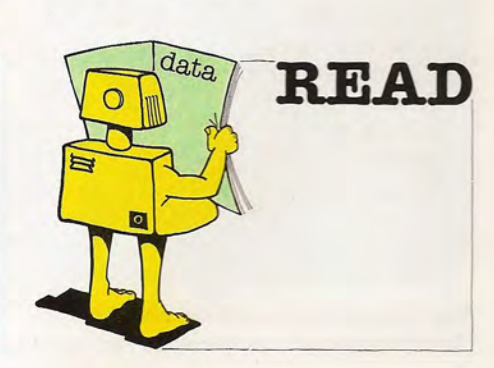
Una vez presentado el formato de ambas instrucciones, cabe analizar su funcionamiento y la forma de utilizarlas dentro de un programa.
Tal como se mencionó anteriormente, si en un programa se utiliza una instrucción READ para la lectura de datos, debe estar presente en el mismo programa al menos una instrucción DATA. Ello resulta imprescindible para que READ pueda leer valores y asignarlos a las variables especificadas en su argumento.
La instrucción READ asignará los valores que lea en la sentencia DATA de la siguiente forma: a la primera variable contenida en READ le asignará el valor de la primera constante que aparezca en la primera instrucción DATA (atendiendo al orden de numeración de las líneas); a la segunda variable, le asignará el segundo valor constante que lea en el correspondiente DATA, y así sucesivamente hasta que quede con asignación la última variable incluida en la instrucción READ.
Veamos un ejemplo:

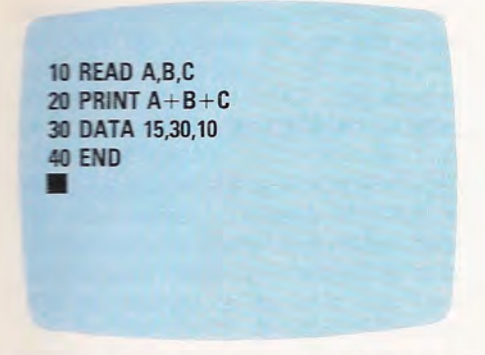
READ

Lee datos de una instrucción DATA y los asigna a las variables especificadas en su argumento.

Formato: (NI) READ <var.> [, <var.>, ..., <var.>]

Ejemplos:
```
10 READ A
20 READ A, B, C
40 READ Z$, DOT$, A
```





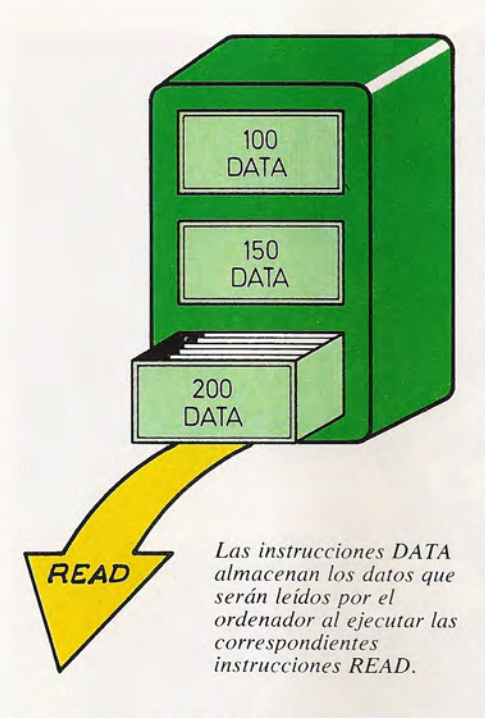
En este caso, la instrucción READ asignará a la variable A el valor 15, a la variable B el valor 30 y a la variable C el valor 10. La ejecución del programa mostrará en la pantalla el siguiente resultado:



Los valores a asignar a las variables que acompañan a READ no tienen por qué estar contenidos en una única instrucción DATA, sino que pueden estar distribuidos en varias. La instrucción READ leerá los valores de la primera DATA hasta que agote sus datos; a continuación, empezará a leer de la siguiente instrucción DATA, prosiguiendo con una tercera sentencia DATA si fuera necesario, hasta completar la asignación. Así, el programa que sigue conduce al mismo resultado que el propuesto en el ejemplo anterior:

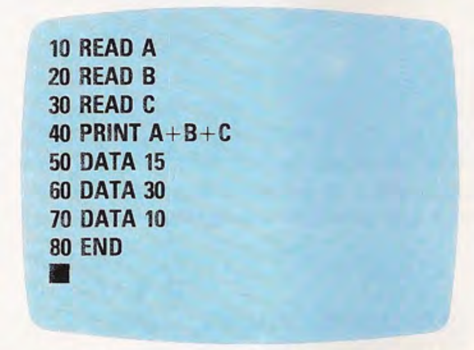
```
10 READ A,B,C
20 PRINT A+B+C
30 DATA 15
40 DATA 30
50 DATA 10
60 END
```

Algo análogo ocurre con las instrucciones READ, ya que su formato es semejante al



Las instrucciones DATA almacenan los datos que serán leídos por el ordenador al ejecutar las correspondientes instrucciones READ.

de las sentencias DATA. En consecuencia, una tercera alternativa para el mismo ejemplo será:



Si el número de variables a asignar con READ, o el número de constantes que hay que aportar, son considerables, será preciso utilizar varias sentencias READ y varias sentencias DATA para cumplimentar las asignaciones.

Las instrucciones READ y DATA pueden estar distribuidas en cualquier zona del programa; si bien, hay que respetar el orden en el que deben ser leídas. Una costumbre generalizada es agrupar a todas las instrucciones DATA para dar mayor claridad al listado del programa, aunque ello no debe considerarse como una regla inquebrantable.

La naturaleza de las variables (numéricas o de cadena de caracteres) a las que se asignará un valor por medio de READ, debe coincidir con la naturaleza (numérica o alfanumérica) de las respectivas constantes incluidas en la correspondiente instrucción DATA. De lo contrario, se producirá un mensaje de "error de asignación" ("type mismatch error", o algo similar), ya que no es admisible el intento de asignar una cadena de caracteres a una variable numérica o viceversa.

```
10 READ A, N$
20 DATA PEPE,35
30 PRINT N$;"tiene"; A;" discos "
40 END
```

La ejecución de las líneas anteriores, producirá en la pantalla el mensaje de error señalado. Para solventarlo, habrá que modificar la línea 10 como sigue:

10 DATA 35,PEPE

Ahora, la ejecución se realizará sin problema alguno:



Una de las ventajas de las instrucciones DATA radica en que para alterar los valo-

DATA

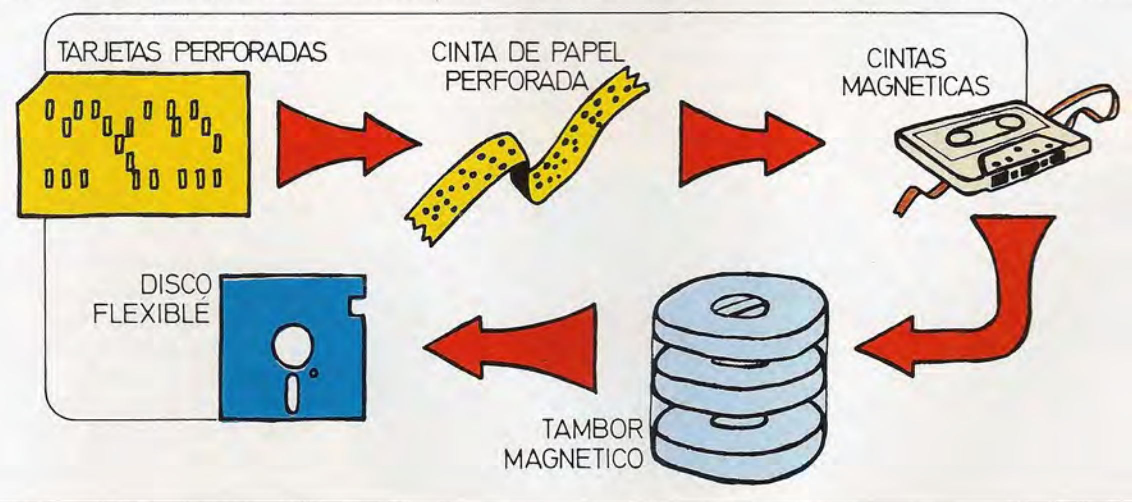
Almacena datos numéricos o cadenas de caracteres para su lectura por medio de instrucciones READ.

Formato: (NI) DATA <const.> [, <const.>, ..., <const.>]

Ejemplos: 10 DATA 1,5,7,30.3,0
60 DATA "SOFT:", 150, PTAS
85 DATA LUNES, 20, ABRIL

Evolución de las unidades de almacenamiento externo

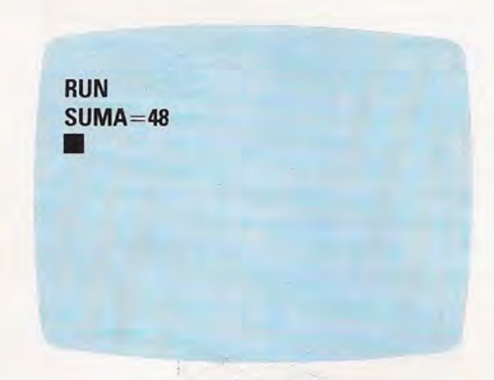
Las memorias de masa o auxiliares son dispositivos periféricos destinados a almacenar de forma permanente grandes volúmenes de información, ya sean programas o datos.
Un programa almacenado en una unidad de este tipo no es directamente ejecutable, sino que, para que ello sea posible, es preciso trasladarlo previamente a la memoria central del ordenador.
Esta categoría de dispositivos presenta una característica fundamental: la información almacenada no es volátil, o lo que es lo mismo, no se borra al desconectar la alimentación. Ello permite conservar los programas y datos para emplearlos en cualquier otra ocasión. Su actuación se fundamenta en el aprovechamiento de las propiedades magnéticas, o en el control de la variación de la estructura física, de determinados materiales que constituirán el soporte de almacenamiento.
Probablemente, el primer soporte empleado para almacenar datos y programas en un ordenador fue la tarjeta perforada. Estas empezaron a utilizarse en la industria textil tras su invención por Joseph Jacquad. Más tarde, las tarjetas perforadas fueron empleadas por Hermann Hollerith en una máquina destinada a procesar la información referente al censo de los Estados Unidos.
Una tarjeta perforada consiste en una simple cartulina rectangular en la que se disponen 12 filas con 80 columnas, cada una de posibles perforaciones. La presencia o ausencia de perforación en los diversos puntos es, precisamente, lo que permite representar distintas informaciones. Para trasladar la información a la tarjeta se emplea comúnmente un código especial, llamado código Hollerith; éste permite codificar cualquier carácter en una columna de la tarjeta.
La perforación de las tarjetas se realiza mediante unas máquinas especiales llamadas perforadoras de tarjetas. Su lectura puede llevarse a cabo por procedimientos mecánicos (escobillas) u ópticos (células fotoeléctricas). En la actualidad, este soporte ha quedado totalmente obsoleto y su presencia se reduce a instalaciones anticuadas de grandes ordenadores.
Las cintas de papel perforado constituyen otro de los soportes de almacenamiento en desuso en nuestros días. Son simples cintas de papel, en las que se practican las perforaciones adecuadas para codificar la información. Hoy en día están presentes en algunos teletipos.
Las propiedades magnéticas de determinados materiales empezaron a aprovecharse para construir los primeros soportes de memoria en cinta magnética. En esencia, la constitución y funcionamiento de las unidades apropiadas no difiere de la de los magnetófonos a casete convencionales. Tampoco el soporte difiere en exceso. Se trata de una cinta plástica sobre la que se deposita una capa de material magnetizable. Esta, que constituye la superficie exterior de la cinta, es la que memoriza la información en forma de dominios magnéticos en distinta orientación. Su tamaño, longitud, formato y capacidad varían según el modelo.
Las cintas magnéticas son soportes de tipo secuencial. Ello supone un inconveniente, puesto que para acceder a una información especifica es necesario pasar por todas las informaciones que la preceden, con la consiguiente pérdida en tiempo de acceso. Semejante problema no es exclusivo de las cintas magnéticas, sino que afecta también a las tarjetas perforadas y a la cinta de papel.
Otro método, también basado en las propiedades magnéticas de algunos materiales, es el de los tambores magnéticos. Consisten en unos cilindros sobre los que se deposita una capa de material magnetizable capaz de retener la información. Esta se graba y se lee mediante un cabezal cuyo brazo se mueve en la dirección del eje de giro del tambor; realizando los giros adecuados del tambor, será posible actuar libremente sobre cualquier punto de la superficie del cilindro. Por consiguiente, el acceso a la información es directo y no secuencial.
En la actualidad, el sistema más extendido lo constituyen los discos magnéticos, ya sean de tipo flexible (floppy disc) o rígidos, además de ciertas variantes menos extendidas. En esencia, se trata de un disco de plástico sobre el que se realiza la consabida operación de depositar una capa de material magnético. Estos discos presentan la ventaja de permitir un acceso directo o aleatorio a cualquier punto de su superficie, ventaja que comparten con el tambor magnético.
Hoy en día se sigue investigando con empeño en este campo, y no sólo para desarrollar nuevos periféricos de almacenamiento —ahí están, por ejemplo, las unidades de disco óptico a láser—, sino también para mejorar la capacidad y eficacia de los discos y cintas magnéticas.



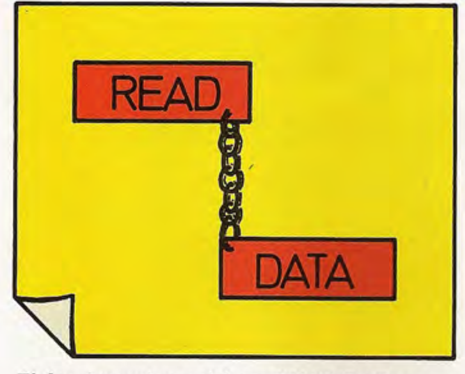
res de los datos del programa, sólo hay que modificar esas líneas; no hay necesidad de cambiar cualquier otra instrucción operativa.
Cuando el número de constantes incluidas en una sentencia DATA no es fijo, sino que en una futura ejecución del programa puede verse ampliado con nuevos datos, es posible recurrir a un sencillo truco. Consiste en introducir un dato especial (distinto de cualquiera de los valores que lo preceden) al final de las constantes de la última sentencia DATA. Este servirá como indicador de que ya se han leído todos los datos anteriores. De esta forma, será posible modificar el número de constantes a leer sin alterar por ello el programa principal. El siguiente ejemplo ilustra la técnica enunciada:

```
10 LET S=0
20 FOR C=0 TO 1 STEP 0
30 READ D
40 IF D=-1 THEN LET C=1
50 IF D< >-1 THEN LET S=S+D
60 NEXT C
70 PRINT "SUMA="; S
80 DATA 10,2,8,23,5,-1
90 END
```

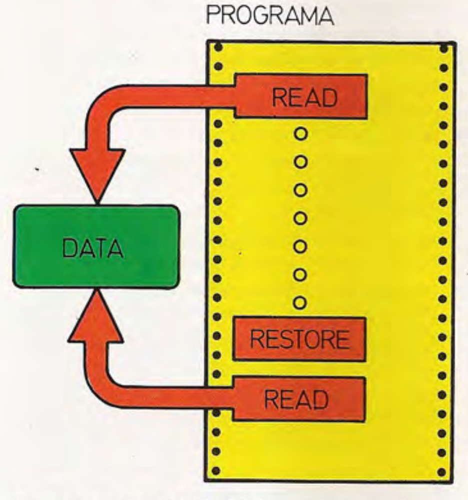
Su ejecución se traducirá en el contenido de la siguiente pantalla:



La instrucción READ irá leyendo los datos de la sentencia DATA, uno por uno, y acumulándolos en la variable S, hasta que llegue al valor −1, el último. Su lectura hará que la variable C que controla al bucle tome el valor 1, con lo que se abandonará el bucle FOR/NEXT y se imprimirá en la pantalla el resultado de la suma de los números aportados en la línea 80.
Si se desea modificar los datos a sumar, o simplemente añadir más números a la lista, será suficiente con alterar la línea 80, sin tocar para nada el resto del programa.



El funcionamiento del par READ/DATA es indisoluble. Cualquier instrucción READ, exige la presencia de una instrucción DATA que aporte los valores a asignar. Estos serán asignados a las variables incluidas en el argumento de la instrucción READ.

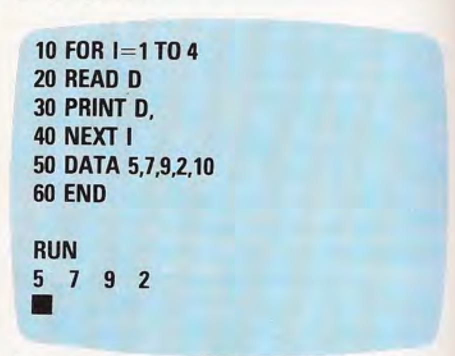


La instrucción RESTORE permite que el argumento de una misma instrucción DATA sea leído y, en consecuencia, utilizado por sucesivas instrucciones READ.

Así, por ejemplo, si se introduce la línea:
80 DATA 10,2,8,23,5,6,4,10,−1
en el lugar de la línea 80 original, la nueva ejecución del programa, obtendrá el siguiente resultado:



El comando READ se puede introducir también de forma directa. No ocurre así con el DATA que, generalmente, debe ser ejecutado de forma indirecta, dentro de un programa.



Si se introduce ahora la orden READ D y, acto seguido, se ejecuta la instrucción PRINT D, aparecerá en la pantalla el número 10. Sin embargo, al ejecutar una vez más la orden READ D, aparecerá en la pantalla un mensaje: "OUT OF DATA ERROR" (error de falta de datos).
Ello se debe a que el ordenador contabiliza interiormente el número de constantes DATA ya utilizadas, y al terminar éstas se ve en la imposibilidad de continuar la lectura.

¿COMO REUTILIZAR LOS VALORES DEL DATA?

¿Qué hay que hacer si se quieren utilizar otra vez, dentro del mismo programa, los datos ya leídos por medio de otras instrucciones READ? El BASIC ofrece como solución el comando RESTORE. Su misión es inicializar la lista de constantes almacenadas en sentencias DATA al primer valor de la primera sentencia DATA, con lo que quedan disponibles de nuevo todas las constantes DATA.
El comando RESTORE se puede introducir tanto de forma directa como indirecta. Cada vez que se utiliza un dato almacenado en una sentencia DATA, se incrementa un determinado registro interno de la máquina. Este registro no es más que un puntero o indicador que señala a la primera constante DATA que aún no ha sido utilizada por el programa. Al ejecutar la orden RESTORE, se borra el mencionado registro, con lo que apuntará al pri-

TABLA DE CONVERSION

ORDENADOR	READ		DATA		RESTORE		
	READ <var.>	**READ <var.>,...,<var.>**	**DATA <const.>**	**DATA <const.>,...,<const.>**	**RESTORE**	**RESTORE <nl>**	**RESTORE <exp.>**
APPLE II (APPLESOFT)	READ <var.>	READ <var.>,...,<var.>	DATA <const.>	DATA <const.>,...,<const.>	RESTORE	?	?
APRICOT (M-BASIC)	READ <var.>	READ <var.>,...,<var.>	DATA <const.>	DATA <const.>,...,<const.>	RESTORE	RESTORE <nl>	RESTORE <exp.>
ATARI	READ <var.>	READ <var.>,...,<var.>	DATA <const.>	DATA <const.>,...,<const.>	RESTORE	RESTORE <nl>	RESTORE <exp.>
CBM 64	READ <var.>	READ <var.>,...,<var.>	DATA <const.>	DATA <const.>,...,<const.>	RESTORE	—	—
DRAGON	READ <var.>	READ <var.>,...,<var.>	DATA <const.>	DATA <const.>,...,<const.>	RESTORE	—	—
EQUIPOS MSX	READ <var.>	READ <var.>,...,<var.>	DATA <const.>	DATA <const.>,...,<const.>	RESTORE	RESTORE <nl>	RESTORE <exp.>
HP-150	READ <var.>	READ <var.>,...,<var.>	DATA <const.>	DATA <const.>,...,<const.>	RESTORE	RESTORE <nl>	RESTORE <exp.>
IBM PC	READ <var.>	READ <var.>,...,<var.>	DATA <const.>	DATA <const.>,...,<const.>	RESTORE	RESTORE <nl>	RESTORE <exp.>
MPF	READ <var.>	READ <var.>,...,<var.>	DATA <const.>	DATA <const.>,...,<const.>	RESTORE	—	—
NCR DM V (MS-BASIC)	READ <var.>	READ <var.>,...,<var.>	DATA <const.>	DATA <const.>,...,<const.>	RESTORE	RESTORE <nl>	RESTORE <exp.>
NEW BRAIN	READ <var.>	READ <var.>,...,<var.>	DATA <const.>	DATA <const.>,...,<const.>	RESTORE	RESTORE <nl>	RESTORE <exp.>
ORIC	READ <var.>	READ <var.>,...,<var.>	DATA <const.>	DATA <const.>,...,<const.>	RESTORE	—	—
SHARP MZ-700 (MZ-BASIC)	READ <var.>	READ <var.>,...,<var.>	DATA <const.>	DATA <const.>,...,<const.>	RESTORE	RESTORE <nl>	RESTORE <exp.>
SINCLAIR QL	READ <var.>	READ <var.>,...,<var.>	DATA <const.>	DATA <const.>,...,<const.>	RESTORE	RESTORE <nl>	RESTORE <exp.>
SPECTRAVIDEO	READ <var.>	READ <var.>,...,<var.>	DATA <const.>	DATA <const.>,...,<const.>	RESTORE	RESTORE <nl>	RESTORE <exp.>
Zx-SPECTRUM	READ <var.>	READ <var.>,...,<var.>	DATA <const.>	DATA <const.>,...,<const.>	RESTORE	RESTORE <nl>	RESTORE <exp.>

<nl>: Número de línea. <var.>: Variable. <const.>: Constante. <exp.>: Expresión numérica con datos y/o variables.

FORMULACIONES DE LOS COMANDOS

READ <var.>: Lee un dato de una instrucción DATA y lo asigna a la variable <var.>. READ <var.>, ..., <var.>: Lee tantos datos como variables haya y los asigna a las respectivas variables. DATA <const.>: Aporta un dato que será leído por medio de una instrucción READ. DATA <const.>, ..., <const.>: Aporta un conjunto de datos que serán leídos por una o varias instrucciones READ. RESTORE: Inicializa el puntero al primer dato de la primera instrucción DATA del programa. RESTORE <nl>: Inicializa el puntero de lectura al primer dato del DATA que ocupa la línea indicada. RESTORE <exp.>: Inicializa el puntero al DATA que ocupa la línea cuyo número resulta al evaluar la expresión.

restore



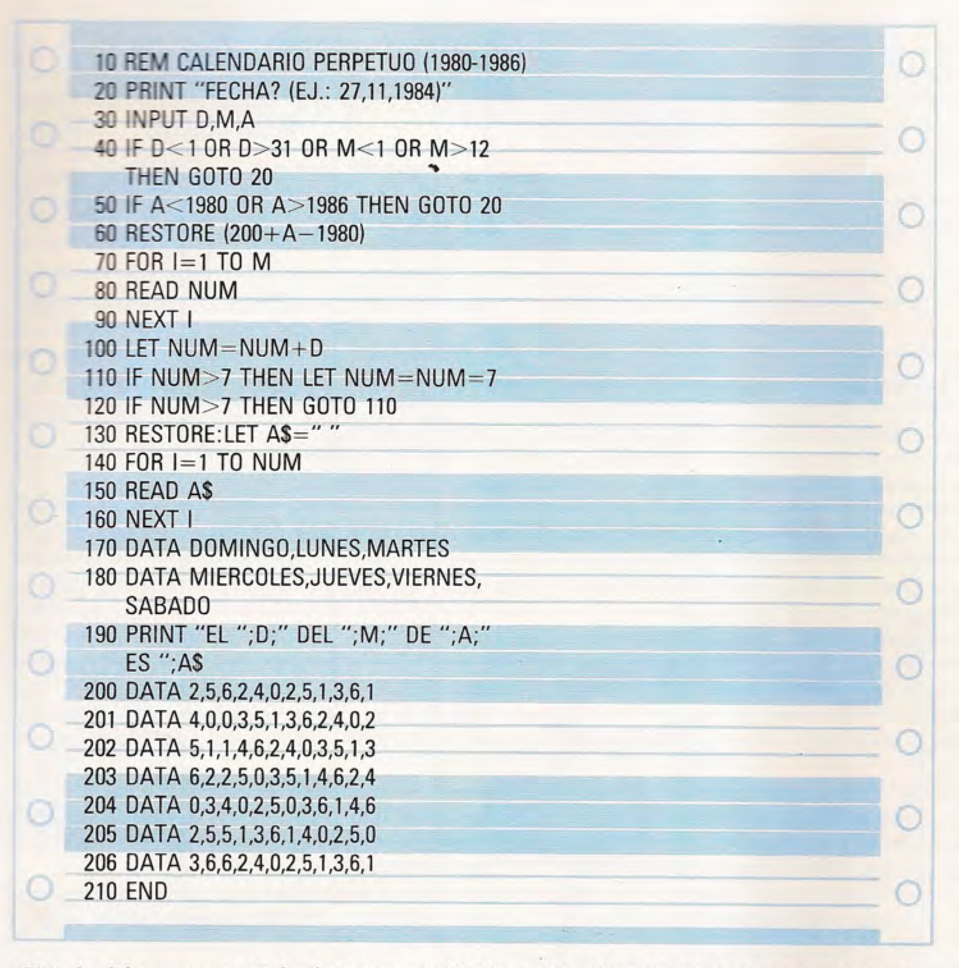
mer dato de la primera sentencia DATA. Algunos dialectos BASIC incluyen una opción muy útil para este comando: permiten especificar un número de línea (o una expresión que calcula un número de línea existente en el programa) detrás de la pa-

RESTORE

Permite que el argumento de instrucciones DATA sea leído de nuevo por medio de instrucciones READ, desde el primer DATA o desde la línea DATA que se especifique.

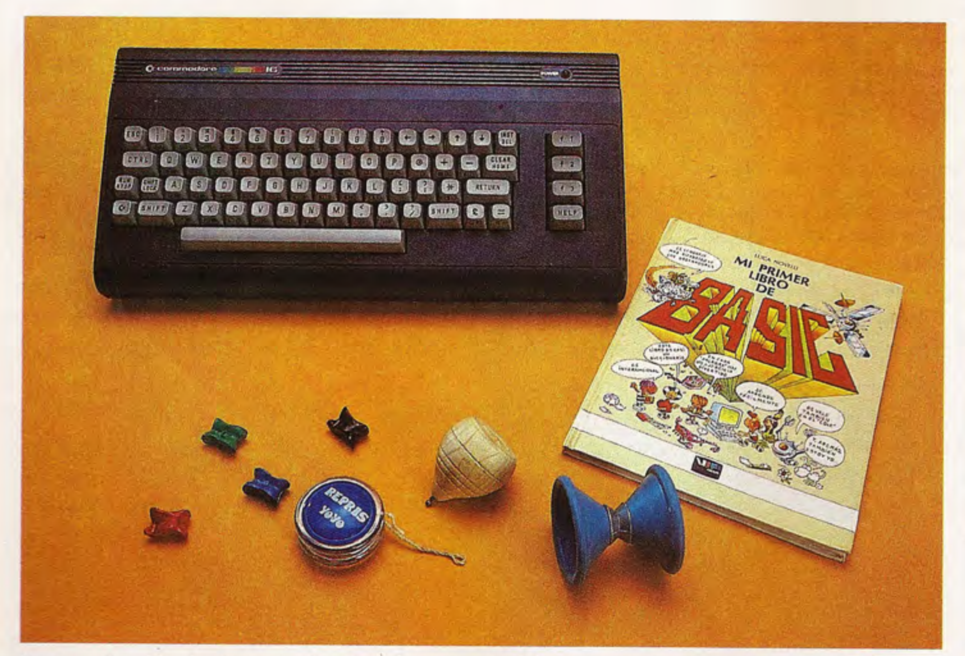
Formato: (Nl) RESTORE [<nl>]

Ejemplos:
```
10 RESTORE
80 RESTORE 30
```



```
10 REM CALENDARIO PERPETUO (1980-1986)
20 PRINT "FECHA? (EJ.: 27,11,1984)"
30 INPUT D,M,A
40 IF D<1 OR D>31 OR M<1 OR M>12
   THEN GOTO 20
50 IF A<1980 OR A>1986 THEN GOTO 20
60 RESTORE (200+A-1980)
70 FOR I=1 TO M
80 READ NUM
90 NEXT I
100 LET NUM=NUM+D
110 IF NUM>7 THEN LET NUM=NUM=7
120 IF NUM>7 THEN GOTO 110
130 RESTORE:LET A$=" "
140 FOR I=1 TO NUM
150 READ A$
160 NEXT I
170 DATA DOMINGO,LUNES,MARTES
180 DATA MIERCOLES,JUEVES,VIERNES,
    SABADO
190 PRINT "EL ";D;" DEL ";M;" DE ";A;"
    ES ";A$
200 DATA 2,5,6,2,4,0,2,5,1,3,6,1
201 DATA 4,0,0,3,5,1,3,6,2,4,0,2
202 DATA 5,1,1,4,6,2,4,0,3,5,1,3
203 DATA 6,2,2,5,0,3,5,1,4,6,2,4
204 DATA 0,3,4,0,2,5,0,3,6,1,4,6
205 DATA 2,5,5,1,3,6,1,4,0,2,5,0
206 DATA 3,6,6,2,4,0,2,5,1,3,6,1
210 END
```

Listado del programa "Calendario perpetuo". Una aplicación práctica de la introducción de datos de la mano de instrucciones READ/DATA.



La introducción de datos en el ordenador a través del programa es una tarea contemplada por el lenguaje BASIC, quien dispone de un grupo de comandos especializados en tal cometido.

labra clave RESTORE. Con ello, en lugar de inicializar el puntero a la primera sentencia DATA, lo hará a la línea DATA indicada (o a la primera que se encuentre a partir de la especificada).

Este refinamiento permite empezar a leer de nuevo los datos de cualquier instrucción DATA, incluso de la última del programa, sin más que especificar su número de línea.

El listado adjunto que corresponde a un programa de *calendario perpetuo* constituye un compendio de los conceptos estudiados en el presente capítulo. Al introducir una fecha determinada a través del teclado, el ordenador determinará automáticamente de qué día de la semana se trata.

La línea 60 ilustra la actuación del comando RESTORE al inicializarlo a un determinado número de línea. En este caso el puntero de datos se inicializa a la línea correspondiente al año introducido, mediante el cálculo de la expresión: (200+a−1980). Su resultado será un número comprendido entre 200 y 206, ambos inclusive. Estos son los números de línea de las sentencias DATA que contienen los valores "mágicos", correspondientes a cada mes de ese año, que permitirán hallar el nombre del día deseado.

Dicho valor, sumado al día del mes, dará un número comprendido entre 1 y 37. A través de las líneas 110 y 120, el número en cuestión se reducirá a un valor comprendido entre 1 y 7 que, como cabe suponer, corresponde al día de la semana comenzando por el Domingo.

Los nombres de los días de la semana están memorizados en dos instrucciones DATA, localizadas en las líneas 170 y 180; éstas se inicializarán con el comando RESTORE de la línea 130. A partir de ahí, ya no queda más que leer el nombre del día con una sentencia READ y almacenarlo en la variable A$ para su posterior presentación en la pantalla:

```
RUN
FECHA/ (EJ.:27,11,1984)
?6,2,1985
EL 6 DEL 2 DE 1985 ES MIERCOLES
■
```

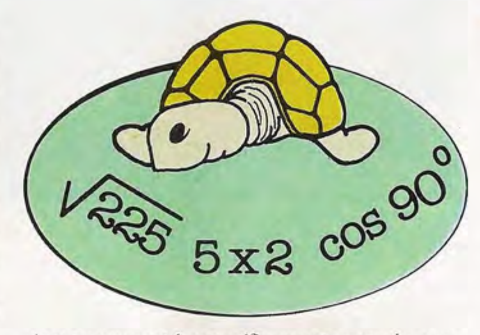
Logo (10)

Operaciones aritméticas y lógicas

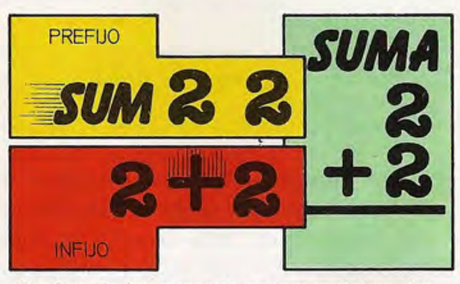
Hasta ahora sólo se han analizado a fondo las capacidades gráficas y de tratamiento de palabras del LOGO. Aunque ambos aspectos son primordiales en este lenguaje, queda aún por comentar otra vertiente fundamental: el tratamiento de datos aritméticos y lógicos. El ordenador es, básicamente, una máquina capacitada para realizar cálculos; en consecuencia, todo lenguaje de programación debe incluir un amplio repertorio de instrucciones aritméticas.
El cálculo numérico no es una posibilidad fácil de explotar en el LOGO. Este lenguaje, al contrario de otros como el FORTRAN o el propio BASIC, no está especialmente orientado al cálculo. No obstante, aporta suficientes funciones aritméticas para realizar la mayor parte de las operaciones habituales.

PREFIJOS E INFIJOS

Las órdenes LOGO pueden ser comandos u operadores. Las funciones aritméticas se encuadran en este segundo grupo. El primer operador aritmético a analizar es el de suma, materializado por medio de la palabra SUM. Como todo operador LOGO tiene sus entradas y salidas. Las entradas de SUM se colocan a la derecha de la orden: SUM 2 3 es un posible ejemplo. Al igual que cualquier otro operador, debe estar precedido por un comando que "recoja" su dato de salida.
Dicha forma de realizar la suma y devolver el resultado se denomina *modo prefijo,* debido a que el operador *precede* a los comandos. Este es el método habitual de utilizar los operadores en LOGO; si bien, puede resultar engorroso en el caso de los operadores aritméticos. Esta es la razón de que exista otra posibilidad, el *modo infijo.* La nueva modalidad consiste en colocar el operador *entre* los operandos. En tal caso, hay que sustituir la palabra SUM por el signo +, por ejemplo: ...2 + 3. Su efecto es idéntico al del ejemplo anterior, aunque su significado es más familiar para el usuario.
Entre los operadores que permiten ambos modos de formulación, se encuentran: SUM (+), PRODUCT (*) y EQUALP (=). Junto a los *prefijos* se indican, entre paréntesis, los correspondientes *infijos.*



Aunque no está específicamente creado para el cálculo matemático, el lenguaje LOGO permite programar operaciones aritméticas con comodidad.



Muchos de los operadores matemáticos del LOGO pueden adoptar una doble formulación: como "prefijo" (precediendo a los datos a operar) o "infijo" (colocados entre los datos de entrada).

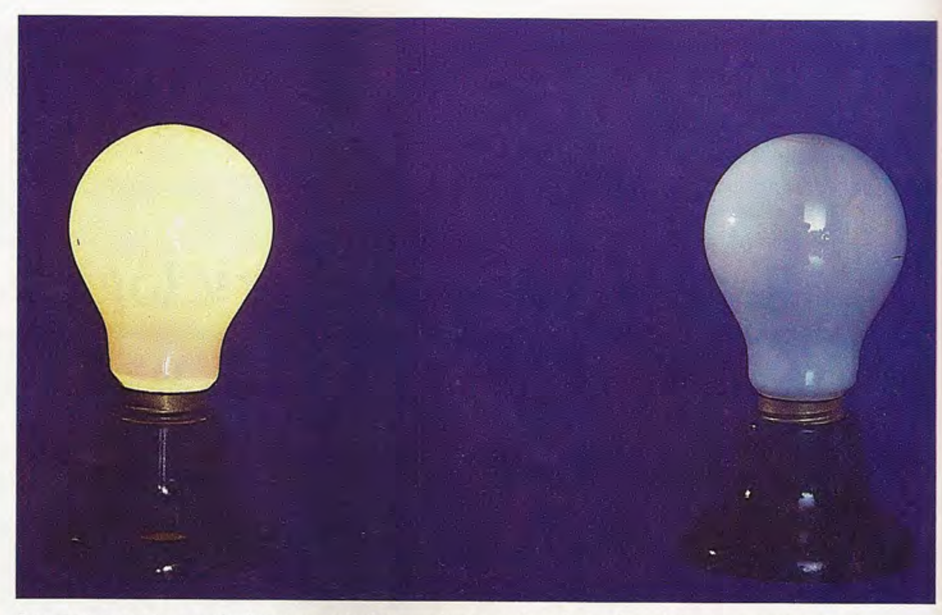
ARITMETICA

Las operaciones básicas del LOGO son las siguientes: suma (+), resta (−), multiplicación (*) y división (/). Todas ellas disponen de un operador infijo (entre paréntesis). Sólo el producto (PRODUCT) y la suma (SUM) admiten prefijo. Hay que señalar que algunos dialectos del LOGO incluyen un prefijo para división, coincidente con la palabra QUOTIENT.
Los operadores SUM y PRODUCT ofrecen, además, la posibilidad de aceptar más de dos datos de entrada. Ello se consigue encerrando el operador y sus datos de entrada entre paréntesis. Por ejemplo: (PRODUCT 2 3 4), cuyo dato de salida será 24 (2 × 3 × 4).
Además de las operaciones elementales, el LOGO incluye algunas funciones matemáticas complejas. Estas son la raíz cuadrada (SQRT) y las funciones trigonométricas seno y coseno (SIN, COS). Todas ellas admiten un solo dato de entrada y, por lo tanto, deben ser formuladas como prefijos. Otra función útil en múltiples casos en REMAINDER. Este operador calcula el resto ocasionado por la división de los datos colocados a su derecha. Es, por lo tanto, un operador en modo prefijo.
Dentro de los operadores aritméticos se pueden considerar los de "truncamiento". Es decir, aquellos que extraen una parte del número indicado. Estos son INT y ROUND. El primero selecciona la parte entera del dato de entrada, mientras que ROUND devuelve el número entero "más cercano" al dato de entrada. Para el dato 15. 7, INT devuelve el valor 15; ROUND, por el contrario, daría como resultado el valor 16.

Todos estos operadores son utilizados, generalmente, para la asignación de valores a una variable. He aquí algunos ejemplos.

```
MAKE "A 2+3
MAKE "SUMA SUM :SUMA :DATO
MAKE "COSA COS :A
MAKE "RESUL INT (PRODUCT 2 :A SIN :A
MAKE "TON 2+:PE/(5-:A)
```

Como puede apreciarse en los ejemplos, es posible encadenar operadores. Cuando los operadores encadenados sean infijos, se evaluarán primero los productos y divisiones y, más tarde, las sumas y restas. En la última línea del ejemplo, se utilizan paréntesis para que el contenido de la variable :PE sea dividido por el resultado de 5 − :A. Si no se coloca el paréntesis, se restaría :A del cociente de :PE/5.



Las comparaciones de datos en LOGO conducen a dos datos de salida de tipo "lógico": TRUE (cierto) o FALSE (falso).

NUMEROS ALEATORIOS

El lenguaje LOGO dispone también de un generador de números aleatorios, muy útil en juegos y simulaciones de azar. El operador RANDOM, seguido por un número entero, entregaría un resultado aleatorio (también número entero) comprendido entre 0 y el dato de entrada. Por ejemplo, RANDOM 5 dará como dato de salida un entero menor que 5.

```
REPEAT 4 (PRINT RANDOM 5)
2
4
3
2
```

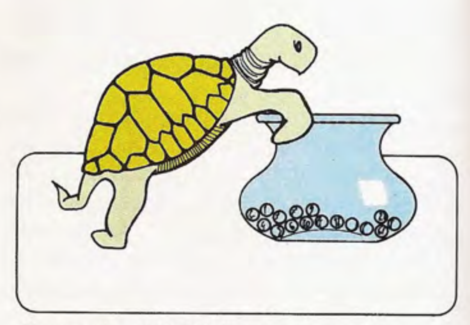
Una determinada secuencia de números aleatorios puede ser repetida por medio de RERANDOM. La instrucción de este comando hace que el ordenador "recuerde" la siguiente secuencia aleatoria generada; cada vez que se vuelve a ejecutar el comando RERANDOM, los números aleatorios generados serán repetidos en el orden en que lo fueron anteriormente. Para la repetición de una secuencia aleatoria, es imprescindible que el argumento de RANDOM coincida con el propuesto en la sentencia recordada. El ejemplo ilustra esta posibilidad.

```
RERANDOM REPEAT 4 (PRINT RANDOM 5)
1
3
4
2

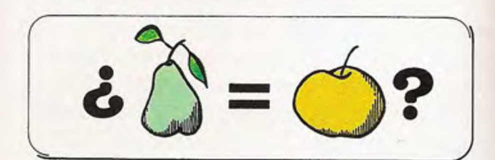
RERANDOM REPEAT 4 (PRINT RANDOM 5)
1
3
4
2
```

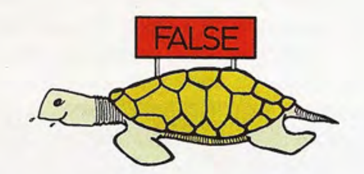
ARITMETICA Y LISTAS

La potencia del LOGO en el tratamiento de listas puede ser aprovechada para fines matemáticos. Esta situación nace de la posibilidad de tratar listas de números. A una lista formada por números pueden aplicársele todos los operadores mencionados en el capítulo de palabras y listas. Por ejemplo, FIRST (1 2 3 4) devolverá el número uno. Este dato de salida puede ser posteriormente procesado por operadores aritméticos. El siguiente procedimiento suma los elementos de una lista de números.



El operador RANDOM genera un número aleatorio, comprendido entre cero y el valor especificado como dato de entrada.





Los operadores de relación sirven para evaluar la certeza (TRUE) o falsedad (FALSE) de una comparación de datos.

TABLA DE COMANDOS LOGO (1)

Instrucción	Cometido	Operador/ Comando
SUM <N1> <N2>	Realiza la suma de los datos de entrada (prefijo)	Operador
<N1>+<N2>	Realiza la suma de los datos de entrada (infijo)	Operador
PRODUCT <N1><N2>	Obtiene el producto de los datos de entrada (prefijo)	Operador
<N1>*<N2>	Obtiene el producto de los datos de entrada (infijo)	Operador
<N1>−<N2>	Opera la resta de los datos entre los que se encuentra (infijo)	Operador
<N1>/<N2>	Divide los datos que acompañan al operador (infijo)	Operador
SQRT <N1>	Halla la raíz cuadrada del dato de entrada	Operador
SIN <N1>	Calcula el seno del dato de entrada	Operador
COS <N1>	Calcula el coseno del dato de entrada	Operador

TABLA DE COMANDOS LOGO (2)

Instrucción	Cometido	Operador/ Comando
RANDOM <N1>	Genera un número entero aleatorio inferior al valor dado	Operador
RERANDOM	Recuerda la última secuencia generada de números aleatorios	Comando
<N1> > <N2>	Entrega el dato lógico TRUE si N1 es mayor que N2	Operador
<N1> < <N2>	Proporciona el dato lógico TRUE si N1 es menor que N2	Operador
EQUALP <N1> <N2>	Proporciona el dato lógico TRUE si N1 y N2 son iguales (prefijo)	Operador
<N1>=<N2>	Proporciona el dato lógico TRUE si N1 y N2 son iguales (infijo)	Operador
AND <L1> <L2>	Realiza la función lógica AND (Y) entre los datos de entrada	Operador
OR <L1> <L2>	Realiza la función lógica OR (0) entre los datos de entrada	Operador
NOT <L1>	Entrega el valor lógico contrario al dato de entrada (NO)	Operador

NOTA: <N1> y <N2> representan datos numéricos.

```
TO SUMA: LISTA
IF: EMPTYP: LISTA(OUTPUT 0) (PRINT SUM
FIRST:LISTA SUMA BUTFIRST:LISTA)
END
```

El significado completo del procedimiento se comprenderá al tratar la recursividad en el capítulo de bucles. Sirva de todas formas este ejemplo para ilustrar la posibilidad de operar con listas de números.

COMPARACIONES

El mundo está lleno de comparaciones, derivadas de las diferencias entre los diferentes individuos. Una persona puede ser más alta que otra, o más obesa. En el terreno matemático estas comparaciones son un calco de las del mundo real. En muchos problemas matemáticos resulta imprescindible determinar si un número x es mayor que otro y. El LOGO aporta tres operadores de relación para la comparación de números: mayor que (>), menor que (<) e igual (=). Los tres actúan como infijos, aunque el tercero admite también la modalidad prefijo; en tal caso su sintaxis es EQUALP. El dato de salida de estos operadores recibe el nombre de dato *lógico,* puede adoptar el valor TRUE (verdadero) o FALSE (falso). En esta tesitura, la salida de 5 > 6 será FALSE mientras que la de EQUALP 5 5 será TRUE.

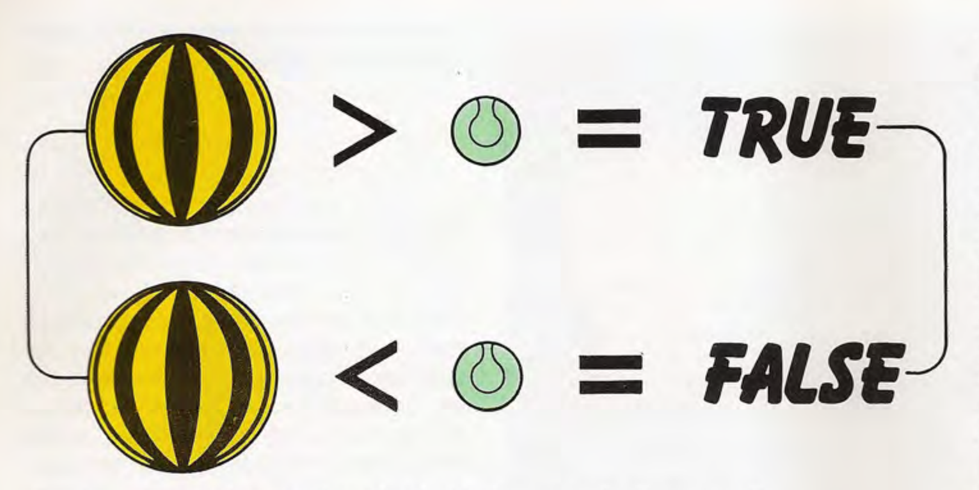
Otros operadores de este tipo —con dato de salida de tipo lógico— han sido estudiados en capitulares anteriores: LISTP y WORDP al hablar de palabras y listas; NAMEP, al hablar de variables: SHOWNP en los TURTLE GRAPHICS...

Los datos de salida de tipo lógico son palabras manejables por el LOGO. De hecho, si se precede un operador de este tipo por un comando PRINT, en la pantalla aparecerá la palabra lógica correspondiente. Por ejemplo: PRINT 5 > 6 escribe en la pantalla la palabra FALSE.

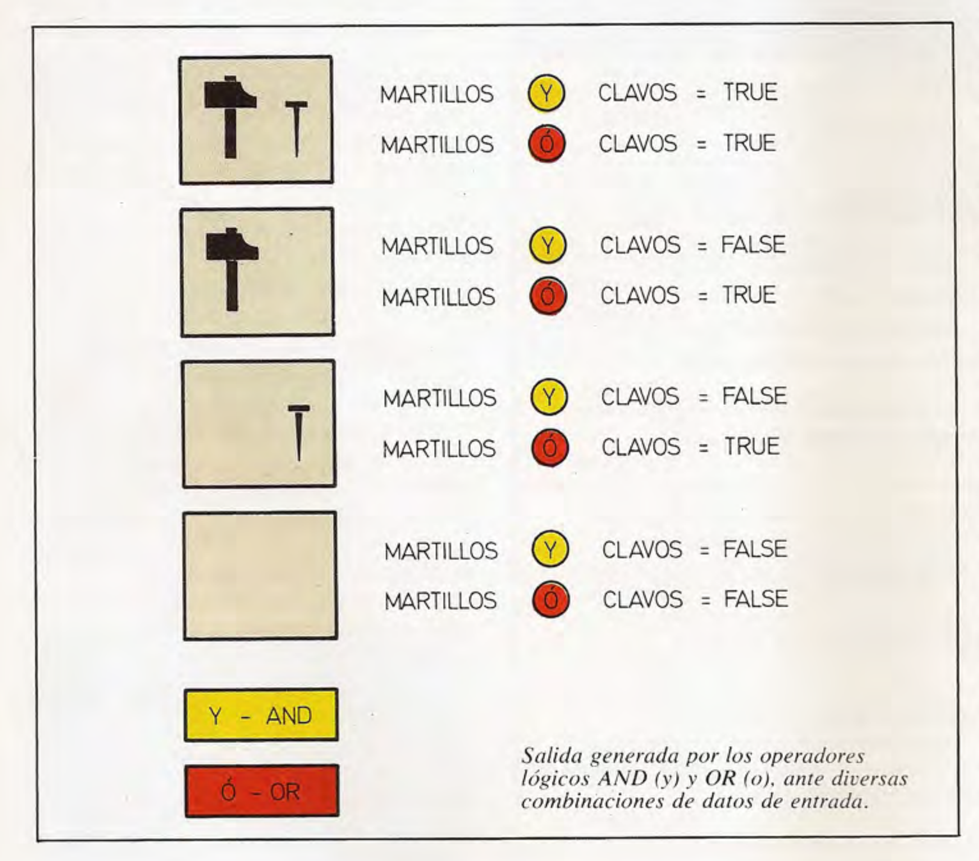
También se pueden utilizar estos datos de salida para formar listas, e incluso procesarlas tal y como se explicó en el capítulo dedicado a palabras y listas. Los datos lógicos tienen además otro significado para el intérprete LOGO. Esta segunda característica de las palabras TRUE y FALSE es la que se detalla en el siguiente apartado.

OPERADORES LOGICOS

Una de las teorías matemáticas más utilizadas en el mundo de los ordenadores es la denominada álgebra de proposiciones o *álgebra de Boole.* Esta parte de la matemática moderna se ocupa de las relaciones entre los enunciados lógicos. En ella existen dos operadores lógicos funda-



Funcionamiento de los operadores de relación "mayor que" y "menor que".



*Salida generada por los operadores lógicos **AND** (y) y **OR** (o), ante diversas combinaciones de datos de entrada.*

OPERADORES LOGICOS AND, OR Y NOT

		AND			**OR**		**NOT**
DATO 1	DATO 2	SALIDA	DATO 1	DATO 2	SALIDA	DATO	SALIDA
TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE
FALSE	TRUE	FALSE	FALSE	TRUE	TRUE		
FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		

mentales AND (y) y OR (o). El cometido de ambos salta a la vista al evaluar la diferencia entre la frase "comeré carne y patatas" y la frase "comeré carne o patatas".

Al tratar con datos de tipo lógico es cuando estos operadores cobran importancia. La instrucción AND :B=0 :C=0 dará el valor TRUE cuando *ambas* variables B y C valgan cero. Realmente, los datos de entrada al operador AND son datos de tipo lógico; el en caso del ejemplo, coinciden con los datos de salida de dos comparaciones.

Si tanto B como C valen cero, el ejemplo anterior equivale a la instrucción AND TRUE TRUE. Por el contrario, si C tuviese un valor distinto de cero, el equivalente sería AND TRUE FALSE.

Estas mismas normas rigen para el uso de OR. La diferencia radica en que, mientras AND exige que sus dos entradas sean TRUE para dar una salida TRUE, a OR le basta con que una de ellas sea TRUE. En el ejemplo comentado al principio, la primera frase será cierta si el individuo come ambas cosas, carne y patatas, y falsa si deja de comer alguna de ellas. Por el contrario, la segunda frase será cierta siempre que coma alguna de las viandas mencionadas y falsa sólo en el caso de que no se coma ninguna de ellas.

Las tablas adjuntas muestran la salida que proporciona cada uno de estos operadores para el conjunto de posibles combinaciones de entrada.

Como quiera que su salida es otro dato lógico, resulta evidente que estos operadores puedan ser encadenados, obteniendo así funciones más complejas. Por ejemplo, la posibilidad de comer "sopa y carne o pescado, o arroz y carne con patatas" se formularía de la siguiente forma:

OR (AND :SOPA OR :CARNE:PESCADO) (AND :ARROZ AND :CARNE:PATATAS).

La certidumbre de dichos enunciados dependerá del valor lógico de las variables :SOPA, :CARNE, etc.

Existe un tercer operador lógico. Este actúa sobre un dato lógico, negándolo, esto es, invirtiendo su valor lógico. El mencionado operador se denomina NOT (no) y su tabla correspondiente es la que aparece en la figura.

Como se puede apreciar en la tabla, el operador NOT admite un único dato de entrada, entregando el valor lógico contrario.

Estos tres operadores son suficientes para relacionar todo tipo de enunciados.

El MP/M en memoria

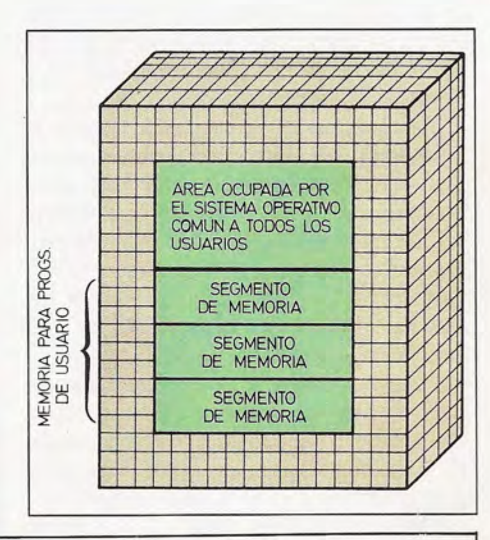
Estructura de la memoria del sistema operativo MP/M

Desde la perspectiva de cualquiera de los múltiples usuarios conectados a un ordenador regido por el MP/M, la máquina se comporta de forma semejante a si se tratara de un equipo monousuario dotado del sistema operativo CP/M. La única salvedad verdaderamente apreciable se encuentra en la disponibilidad de un conjunto de comandos adicionales; éstos permiten gestionar las transferencias de información de una forma distinta, o con nuevas posibilidades debido a la existencia de varios usuarios que comparten los recursos del sistema.

El número máximo de usuarios que admite el sistema operativo MP/M es de 16; numerados, respectivamente, del 0 al 15. Ello impone, obviamente, una serie de cargas sobre la memoria principal del sistema; toda vez que cada usuario recibe la asignación automática de una porción de la misma cuando se conecta al sistema.

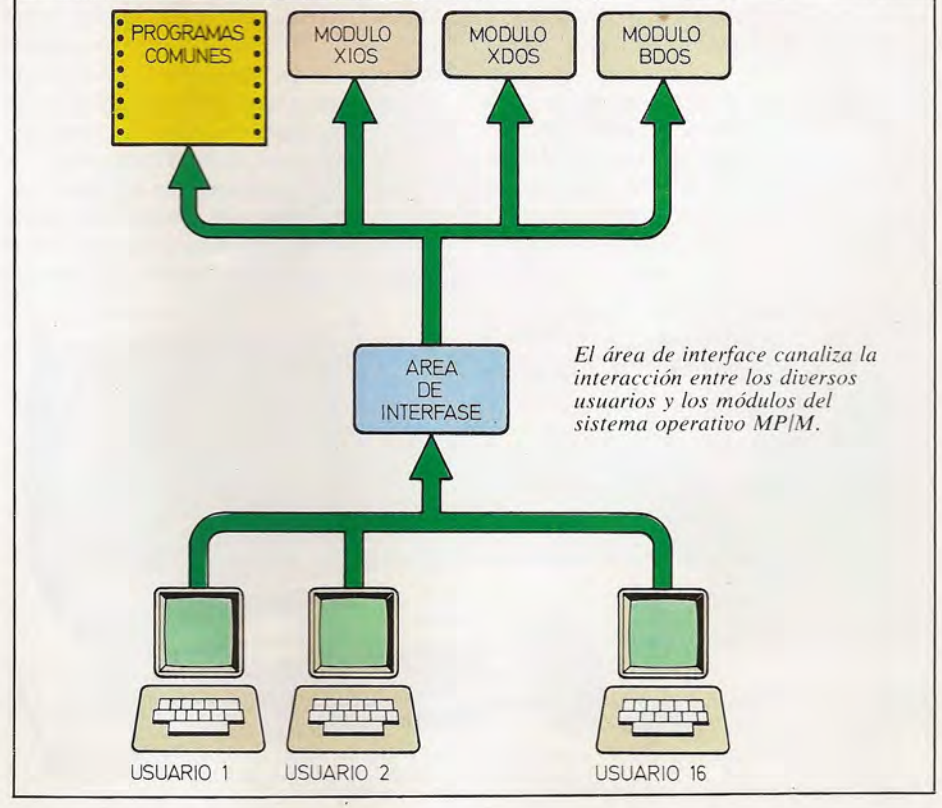
Es evidente, que el espacio total de memoria ha de ser lo bastante grande para que los usuarios no se vean excesivamente limitados en el uso de programas de relativo tamaño. Por lo que respecta a las exigencias hardware, el sistema MP/M necesita un mínimo de 48 Kbytes de memoria RAM en el ordenador que lo soporta. Hay que tener en cuenta que sus posibilidades le permiten controlar hasta dieciséis unidades de disco asociadas al equipo. El espacio de memoria ocupado por el sistema operativo es variable; por un lado depende de la propia versión del sistema operativo que se utilice, y por otro de la configuración física de terminales y periféricos. Acerca de este último condicionante, cabe precisar que hay una serie de informaciones, imprescindibles para el funcionamiento del sistema, que corresponden a cada usuario y a cada consola; informaciones que, por supuesto, requieren un espacio de almacenamiento en la memoria central del ordenador.

Distribución de la memoria central del ordenador al actuar bajo el control del sistema operativo MP/M.



ESTRUCTURA DE LA MEMORIA PRIMARIA

Al cargar el sistema operativo MP/M en la memoria primaria del ordenador, desde el



El área de interface canaliza la interacción entre los diversos usuarios y los módulos del sistema operativo MP/M.

disco en el que se encuentra almacenado y por efecto de un proceso de arranque en frío, éste pasa a ocupar una determinada zona de la memoria de uso exclusivo. El resto de la memoria primaria queda destinada al almacenamiento de los programas propios de los distintos usuarios que acceden al sistema.

La ubicación en la memoria física del conjunto de módulos que constituyen el MP/M es muy similar a la propia del CP/M, aunque en algunos aspectos existen notables diferencias; principalmente en el área de programas de usuario. Esta última se distribuye internamente en una serie de particiones denominadas *segmentos de memoria*.

En la zona alta de la memoria se aloja el módulo XDOS (Extended Disk Operating System); aunque no necesariamente en la posición superior de dicha zona, ya que el MP/M permite declarar una serie de programas residentes en memoria, y por lo tanto, utilizables por todos los usuarios del sistema, sin necesidad de repetir su presencia en la zona reservada a cada usuario. De producirse tal situación, la porción más alta de la memoria se destinaría a estos programas, y a partir de ellos quedaría emplazado el módulo XDOS. Tras éste, y en dirección a la zona baja de la memoria, se encontraría el módulo BDOS (Basic Disk Operating System) y, a continuación, el módulo XIOS (Extended Input Output System).

Hasta este nivel, la estructura es similar a la que corresponde al sistema operativo CP/M. De inmediato aparece un área de memoria, propia del MP/M, cuyo objeto es permitir la interacción entre el usuario y el sistema operativo. Esta se subdivide internamente en tres zonas:

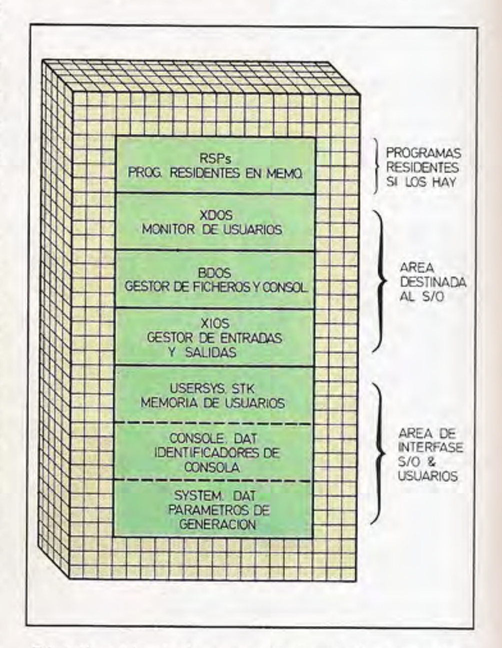
USERSYS.STK
CONSOLE.DAT
SYSTEM.DAT

La porción de memoria ocupada por USERSYS.STK es de tamaño variable, ya que depende del número de segmentos en los que se ha dividido el área de memoria destinada a usuarios. Exactamente son 64 bytes de memoria los que se asignan a cada segmento, a utilizar como área de almacenamiento temporal para que los programas de usuario hagan sus llamadas al sistema operativo (CALL BDOS).

El área destinada a CONSOLE.DAT también es de amplitud variable, dado que depende del número de consolas asociadas a la CPU. Cada consola conectada requiere un total de 256 bytes; éstos memorizan lo que se conoce por las siglas TMP (Terminal Message Process): programa reentrante en memoria por medio del cual se transmiten los comandos introducidos a través del terminal a las colas de órdenes. En esta misma zona se almacenan un conjunto de indicadores que sirven para identificar la consola en las operaciones de entrada/salida.

El área de memoria destinada a SYSTEM.DAT tiene una longitud dependiente del número de segmentos en los que se divide la memoria destinada al usuario. Por cada segmento, son necesarios 256 bytes en la zona SYSTEM.DAT. En este espacio de memoria se almacena la información necesaria para la reconfiguración dinámica del sistema dependiendo de las necesidades de utilización en cada momento.

El resto de la memoria, en su zona baja, queda disponible para los usuarios. Dada la naturaleza multiusuario del MP/M, lo habitual es que dicha sección de memoria esté ocupada simultáneamente por varios programas. Para favorecer su almacenamiento, la referida zona estará fraccionada en nuevas subdivisiones, hasta un máximo de ocho, denominadas segmentos de memoria. Su tamaño, definido durante

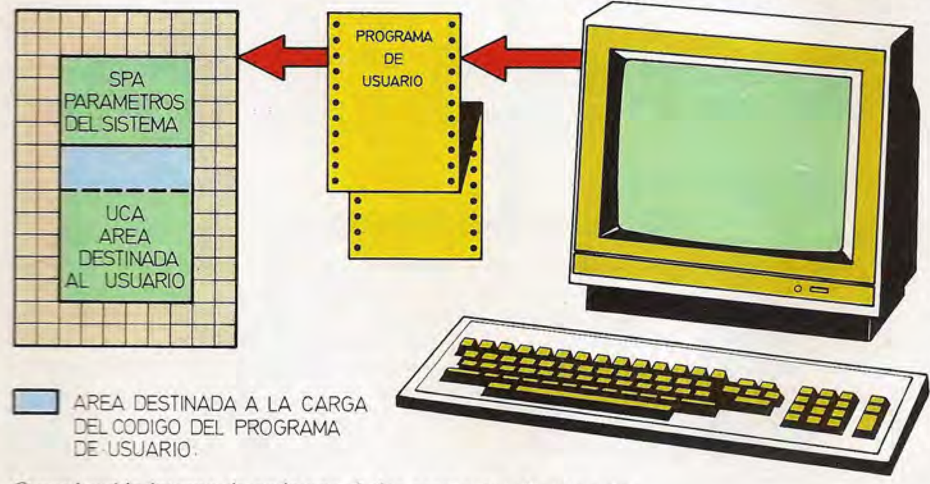


Distribución de la zona de memoria central común a todos los usuarios del sistema.

la generación del sistema, es variable entre 0 y 64 Kbytes.

La generación del sistema consiste en el ajuste de toda una serie de parámetros, de acuerdo con las necesidades del usuario, en orden a que la máquina opere de forma adecuada y sin emplear recursos innecesariamente. Este es un proceso de total relevancia cuando el ordenador en cuestión no tiene una capacidad muy elevada; circunstancia muy común entre los microordenadores equipados con el sistema operativo MP/M.

Cada uno de los segmentos destinados al usuario es comparable al área de TPA (Transient Program Area) del sistema operativo CP/M. A su vez, cada segmento se divide en dos partes SPA (System Parameter Area) y UCA (User Code Area). La porción SPA ocupa los primeros 256 bytes de cada uno de los segmentos y contiene, como su propio nombre indica, determinados parámetros del sistema. En ocasiones, recibe el nombre de "página base del sistema", ya que en el sistema MP/M, al igual que sucede en el CP/M, el tamaño de la página de memoria es de 256 bytes.



Organización interna de cada uno de los segmentos de memoria.

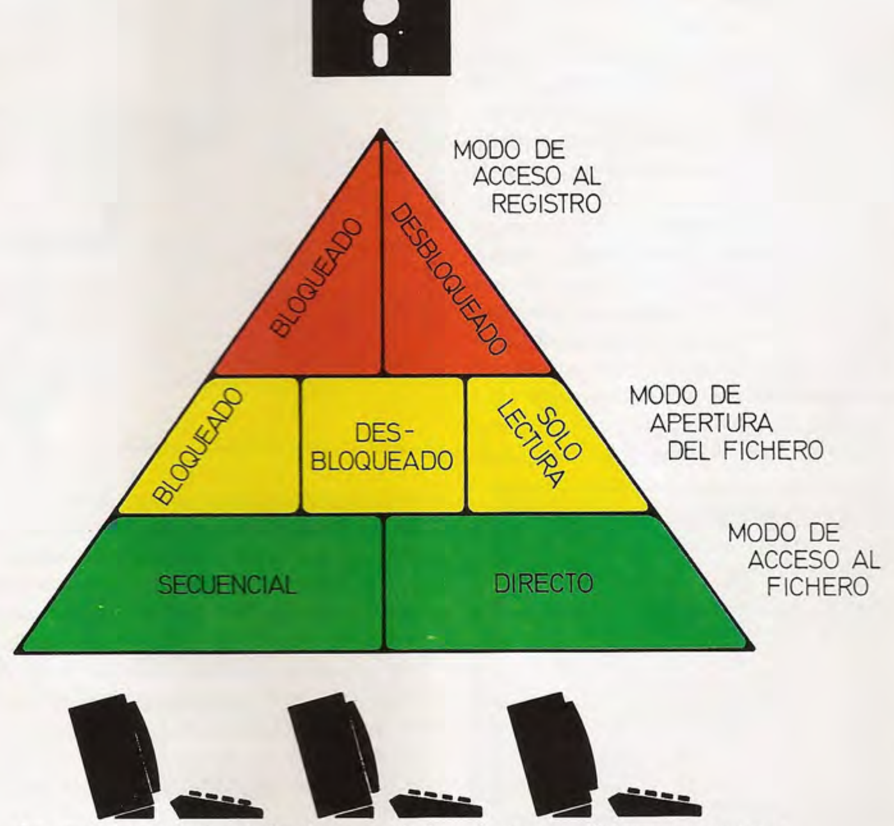
El resto del espacio de memoria propio de cada segmento constituye el área UCA. Cuando un programa es cargado en memoria, su código objeto se carga al principio del área UCA; la operación de carga la realiza el intérprete de comandos CLI (Command Line Interpreter) quien, además, es el encargado de gestionar la comunicación entre las colas y el sistema operativo.
A raíz de lo descrito, queda patente que la zona de memoria ocupada por los archivos propios del sistema operativo, es compartida por todos los usuarios; el acceso a la misma se gestiona a través del área de "interface" o de interacción con el usuario.
El hecho de que no existan repeticiones

Acceso concurrente a ficheros

Al entrar en un entorno multiusuario, la gestión de los ficheros se complica: varios usuarios pueden intentar un acceso simultáneo, dando origen a conflictos en cuanto a la veracidad y seguridad de la información del sistema.
Un caso muy frecuente es el que se presenta cuando varios usuarios intentan actualizar un fichero y se mandan dos órdenes de actualización, casi simultáneamente, hacia el mismo registro. En esta situación, actualizaría el registro en primer lugar del usuario que ejecutó la orden de modificar el registro con mayor antelación. No obstante, puede ocurrir que antes de concluir la modificación dirigida por el primer usuario, empiece a ser activa la del segundo. Ello conduce a una situación conflictiva, ya que se están intentando dos operaciones de modificación simultáneas sobre un mismo registro.
Al actuar sobre un fichero, ya sea de tipo secuencial o de acceso aleatorio (acceso directo), el sistema elude los posibles inconvenientes con el uso combinado de diversos modos de apertura de ficheros y con el bloqueo de los registros a modificar.
Cuando se procede a la apertura de un fichero se pueden especificar tres modos de actuación:

— Modo bloqueado.
— Modo desbloqueado.
— Modo de sólo lectura.

Con apertura en *modo bloqueado,* un proceso sólo puede abrir un fichero si éste no estaba ya previamente abierto. Una vez que se ha realizado su apertura ningún otro proceso puede acceder a dicho fichero hasta que el proceso inicial proceda a cerrarlo. Los ficheros abiertos de esta forma pueden ser objeto de lectura y escritura, a no ser que, además, se haya indicado que el fichero es accesible sólo para lectura, en cuyo caso ésta será la única operación realizable.
La apertura en *modo desbloqueado* permite abrir un fichero aunque otro proceso lo haya hecho con anterioridad; en tal caso, todos los procesos que abran el fichero pueden acometer operaciones de lectura y escritura.
En *modo de sólo lectura,* el intento de apertura de un fichero por parte de un proceso, sólo resultará eficaz si el fichero no estaba ya abierto por otro proceso; o si, aún estando abierto, tal operación se ha realizado también en modo de sólo lectura. Como su nombre indica, en esta situación sólo se permitirá la lectura del fichero, resultando prohibida la modificación de cualquier parte del mismo.
A raíz de los párrafos anteriores, se observa que aún no se ha resuelto el problema de la concurrencia sobre un registro, aunque se han puesto trabas para que no todos los procesos tengan la capacidad de modificar un fichero. En consecuencia, es necesario bloquear los registros, además de los ficheros, para evitar posibles contingencias.
El bloqueo de registros sólo puede realizarse sobre ficheros abiertos en modo bloqueado o de sólo lectura. Tal medida hace posible que al ejecutar una operación que modifique el fichero, el proceso sea dueño exclusivo del registro, bloqueando el acceso de los demás procesos; una vez concluida la transacción pendiente, el registro quedará liberado.
Estas dos herramientas son utilizadas por el módulo BDOS del sistema operativo para gestionar las entradas y salidas concurrentes a ficheros.



La puesta en práctica de las distintas técnicas de acceso a ficheros y a registros, permiten la explotación concurrente de los ficheros del MP/M.

El MP/M es un sistema operativo con capacidad para trabajar en régimen multiusuario y multitarea.



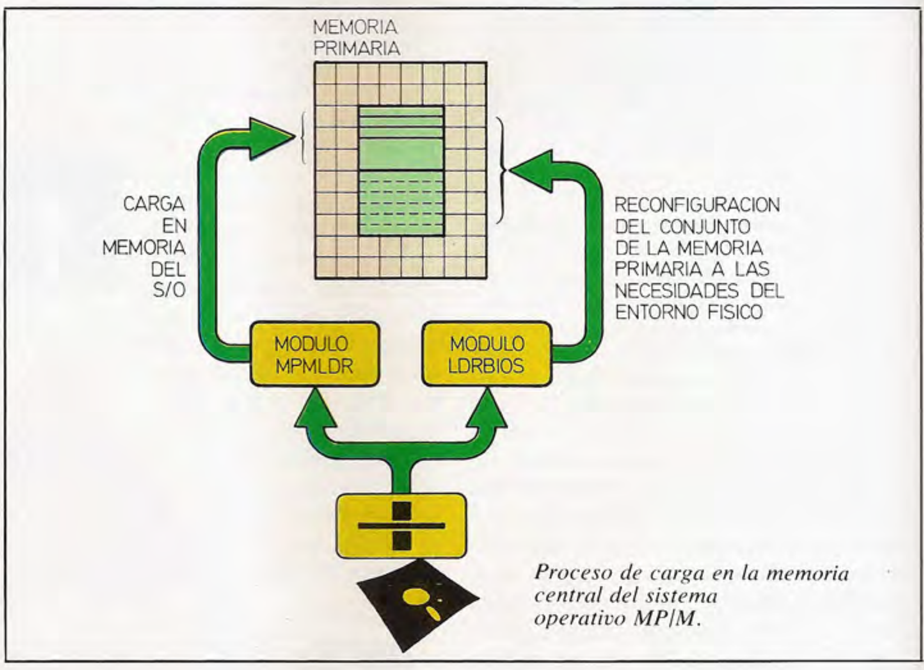
de información o duplicidades en el almacenamiento de los archivos del sistema operativo favorece la dedicación de un mayor espacio de memoria a los usuarios.

ESTRUCTURA DEL DISCO

En un párrafo precedente se ha señalado que antes de la carga del sistema operativo MP/M, es necesario proceder a la definición de una serie de parámetros necesarios para su eficaz operación; parámetros que se concretan, por ejemplo, en el número y tamaño de los segmentos en los que es preciso dividir el área de memoria destinada a los usuarios. La generación puede realizarse empleando determinados comandos, almacenados en la memoria permanente (ROM), o bien por medio de un programa especial que se carga en la memoria con este fin exclusivo.

Al igual que ocurre con el sistema operativo CP/M, el MP/M ocupa las dos primeras pistas del disco sobre el que se encuentra almacenado. Sin embargo, dado que es bastante más voluminoso, su presencia se extiende a otras pistas del disco. Más exactamente, el MP/M se almacena adoptando la forma de un fichero denominado "MPM.SYS"; éste constituye la imagen del sistema operativo MP/M en memoria primaria. Las dos pistas reservadas, la 0 y la 1, se encuentran ocupadas por el cargador del sistema operativo. El cargador, a su vez, se encuentra dividido en dos fragmentos o módulos, denominados, respectivamente, MPMLDR y LDRBIOS. Ambos fragmentos se comportan como programas independientes, organizando la carga en memoria de los módulos constitutivos del MP/M y organizando los datos del área de "interface" usuario-sistema. El MPMLDR utiliza datos que son generados previamente por el módulo LDRBIOS. Estos programas son activados por el módulo "bootstrap", localizado en la pista cero sector 1 del disco.

El módulo LDRBIOS es el cargador que depende del entorno físico de la máquina; esto es, del número y tipo de los periféricos conectados y de la interrelación entre dispositivos físicos y lógicos. Sus primitivos, situados en posiciones sumamente determinadas, son empleados por el módulo de carga MPMLDR. Su complementario, el módulo MPMLDR, constituye el cargador lógico del sistema operativo MP/M. Es independiente del entorno físico, ya que su misión se reduce a tomar el fichero MPM.SYS y trasladarlo a la memoria primaria con la ayuda del módulo de carga LDRBIOS; por supuesto, tal operación la realiza adaptando la estructura del fichero a la que ha de tomar el sistema operativo en la memoria.



Proceso de carga en la memoria central del sistema operativo MP/M.

PISTA	SECTOR	MODULO
00	01	BOOT
00	02	MPMLDR
00	25	
00	26	LDRBIOS
01	26	

Distribución de las pistas del disco reservadas al sistema operativo MP/M.

Visicalc (2)

Funciones para el cálculo de los elementos de la hoja electrónica

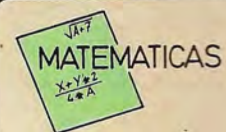
Tras el estudio de los principales comandos que ofrece el VISICALC para gestionar la hoja electrónica, quedan por analizar las funciones cuyo cometido es definir las fórmulas de cálculo. A pesar de la gran importancia que adquieren los comandos, ya que de ellos depende la comodidad del trabajo del usuario en la pantalla, tal vez sea aún superior la relevancia de las funciones; éstas constituyen el instrumento apropiado para el cálculo automático de los elementos de la hoja electrónica.

FUNCIONES DEL VISICALC









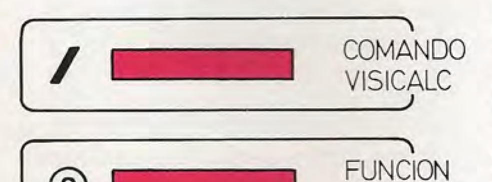
Las funciones que brinda la hoja electrónica VISICALC son catalogables en cuatro grupos genéricos: financieras, lógicas, especiales y matemáticas. Algunas funciones pueden adscribirse a dos grupos; tal es el caso de la función "máximo", encuadrable como función matemática o financiera.

El símbolo reservado para identificar a las funciones VISICALC es "@". Ello permite, tanto al usuario como al propio programa, diferenciar cómodamente el contenido de un elemento: si no va precedido por ningún símbolo especial, se tratará de un dato numérico o alfabético; en caso contrario, éste será un comando si comienza por el símbolo "/", o una función si es "@" el símbolo que lo precede.

A continuación se describen las principales funciones manejadas por el VISICALC, detallando, en cada caso, los posibles argumentos y su significado.

FUNCIONES FINANCIERAS

Uno de los principales objetivos de las hojas electrónicas es facilitar el trabajo de ejecutivos comerciales, economistas y financieros. Las funciones agrupadas en este epígrafe resultan especialmente adecuadas para este cometido.



La distinción entre comando y función VISICALC corre a cargo del símbolo identificador que precede a cada elemento.

AVERAGE

Sirve para calcular la media de un conjunto de valores contenidos en elementos de la matriz. El máximo número de elementos que se pueden incluir en el argumento de esta función se eleva a 256. Su formato es:

@ AVERAGE (lista de elementos).

COUNT

Se utiliza para contar el número de elementos de la lista que constituye su argumento. El formato para invocar a la referida función es:

@ COUNT (lista de elementos).

NPV

Aun encabezando el grupo de funciones financieras, las dos funciones anteriores resultan útiles para otros cálculos de tipo general; en cambio, la función NPV es típicamente financiera. Como argumento, debe contar con una expresión o un valor concreto (P), y una porción de una fila o columna de la matriz; este segundo argumento se especifica mediante la coordenada inicial, la final y un valor de "paso" para llegar de una a otra.

A partir de estos argumentos, la función NPV devuelve el valor neto actual del "cash flow" de los elementos indicados, descontando el porcentaje P. El formato de esta función es:

@ NPV (porcentaje de descuento, parte de una línea).

LOOKUP

Utiliza como argumento una expresión o valor concreto designado por X, y una zona delimitada de una fila o columna. Su actividad consiste en extraer el máximo valor de la serie de elementos aportados que sea inferior a X. Por lo demás, almacena ese valor en la posición situada a la derecha y debajo de la línea de entrada. Su formato de llamada es:

@ LOOKUP (tope de búsqueda, parte de una línea).

CHOOSE

Sirve para localizar el n-ésimo elemento de valores de entrada. Para ello, utiliza como parámetros el cardinal buscado y una lista cualquiera de elementos de la matriz. Su formato coincide con:
@ CHOOSE (número de orden, lista de elementos).

MAX

Permite localizar el máximo valor dentro de un conjunto de elementos de entrada. Su único argumento se reduce a la lista de elementos entre los que se debe localizar el valor máximo; de ahí que su formato sea:
@ MAX (lista de elementos).

MIN

Función análoga a la anterior, aunque localizando esta vez el mínimo valor que aparece entre los elementos de entrada. Su formato es:
@ MIN (lista de elementos).

SUM

Calcula el sumatorio de los valores contenidos en la lista de elementos que constituye su único argumento. Para invocar a esta función hay que recurrir al siguiente formato:
@ SUM (lista de elementos).

FUNCIONES LOGICAS

Una característica común a todas las funciones lógicas que se detallan a continuación, es que tanto los parámetros de entrada como el resultado, sólo admiten elementos cuyos valores sean *"verdadero"* o *"falso"*.

AND

La función lógica AND (Y), produce como resultado el valor *"verdadero"* si y sólo si todos sus argumentos de entrada son *"verdaderos"*. Los parámetros de entrada para esta función pueden ser valores lógicos (*"verdadero"* o *"falso"*), o bien expresiones conformadas con valores lógicos. El formato de llamada es:
@ AND (lista de elementos lógicos).

FALSE

En este caso se trata de una función constante que produce siempre como resultado el valor *"falso"*. Por lo tanto, cabe afirmar que la función FALSE está exenta de parámetros de entrada:
@ FALSE.

IF

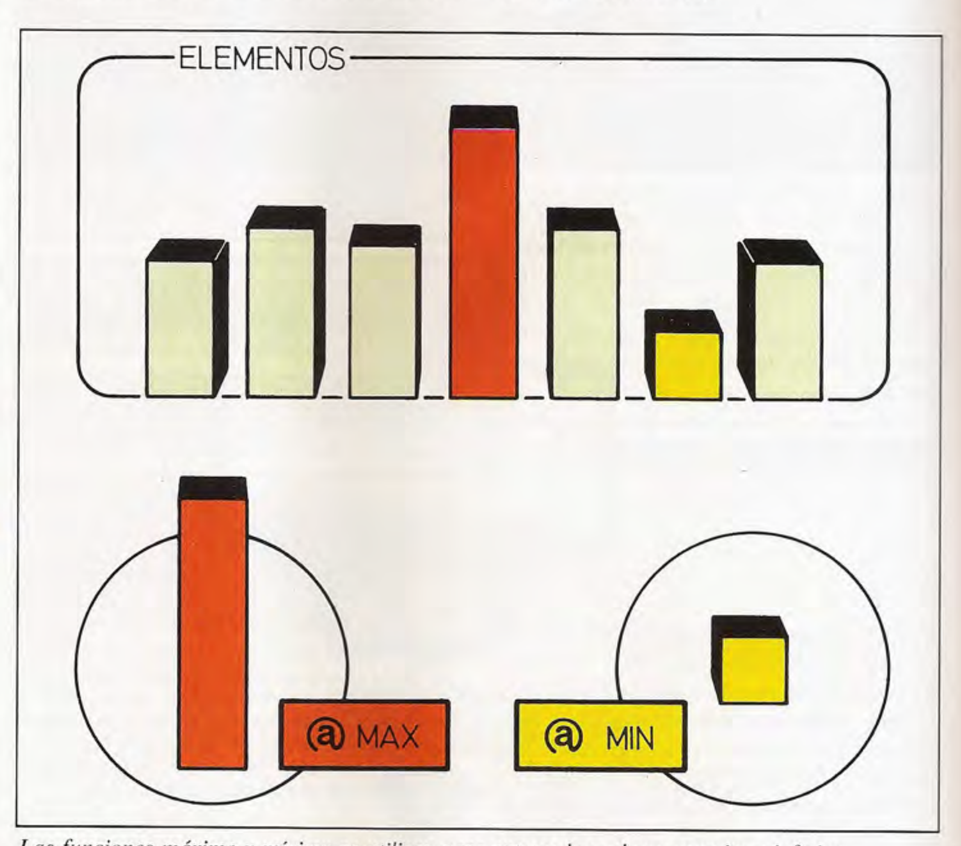
Aun siendo de tipo lógico, la función IF admite como parámetros de entrada valores numéricos o literales; ello constituye una excepción dentro de las funciones lógicas. Como argumento de entrada admite una expresión de cualquier tipo (por ejemplo, elemento 1−B<5), y dos valores cualquiera (por ejemplo, X e Y). El resultado producido por la función será X en el caso de que la expresión sea *verdadera,* y coincidirá con Y cuando la expresión sea *falsa.* La forma de llamar a esta función es:
@ IF (expresión, pareja de valores).

ISERROR

Esta función admite a un elemento de la hoja electrónica como único argumento. Si el valor de dicho elemento es *"ERROR",* el resultado producido por la función será verdadero; en el caso contrario, esto es, si el elemento adopta cualquier valor distinto de *"ERROR",* el resultado será *falso.* Su formato coincide con:
@ ISERROR (elemento).

ISNA

La finalidad de esta función es semejante a la anterior; si bien el criterio para producir el resultado será *"NA"* en vez de *"ERROR"*. Por lo tanto, el resultado de la función será *verdadero* si el valor de su único argumento es *"NA"* y será *falso* si el valor del argumento es distinto de *"NA"*:
@ ISNA (elemento).



Las funciones máximo y mínimo se utilizan para extraer los valores superior e inferior, respectivamente, del conjunto de elementos incluidos en su argumento.

NOT

La función lógica NOT (NO) devuelve un resultado lógico opuesto al de su único argumento de entrada. En efecto, si el argumento tiene como valor *verdadero,* el resultado producido por la función NOT será *falso,* mientras que si el valor del argumento es *falso,* el resultado producido será *verdadero.* El formato de llamada es el siguiente:
@ NOT (elemento).

OR

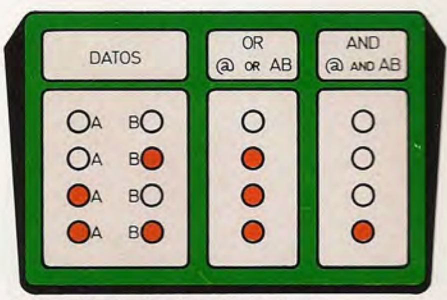
Corresponde a la función de suma lógica.

De ahí que para producir un resultado *verdadero,* sea suficiente con que uno solo de los elementos lógicos de entrada sea *verdadero,* con independencia de los valores que puedan tomar los restantes argumentos:
@ OR (lista de elementos lógicos).

TRUE
De nuevo se trata de una función constante. No precisa ningún argumento de entrada, ya que, invariablemente, produce como resultado el valor lógico *verdadero.* El formato adecuado es:
@ TRUE.

FUNCIONES ESPECIALES

Dos de las funciones lógicas descritas tenían encomendada la misión de comprobar si el contenido de un elemento era "ERROR" o "NA" (el significado de "ERROR" es obvio y el de "NA" es "no disponible", o *"not available"* en inglés). El VISICALC ofrece dos funciones especiales para conseguir que algún elemento de la hoja llegue a contener alguno de los referidos valores.



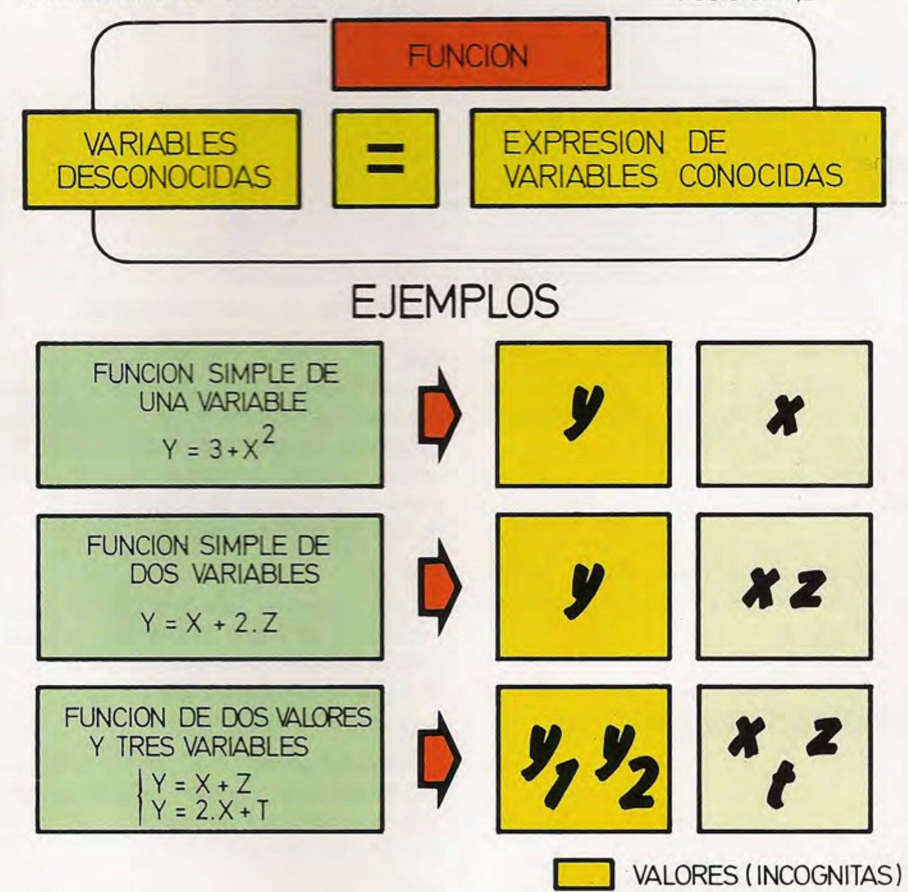
Actuación de las funciones lógicas AND y OR.

ERROR
Dependiendo de la entrada producida,

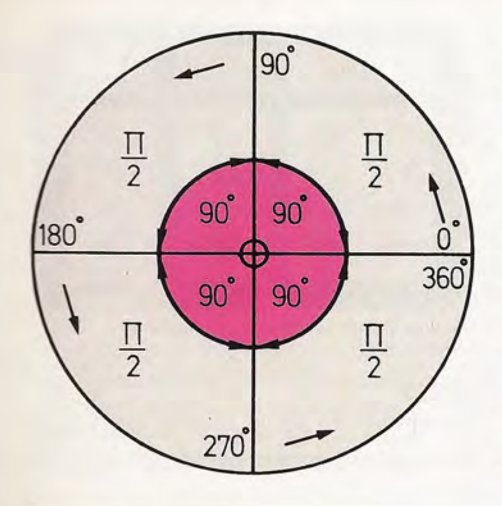
Teoría de funciones (I)

Para manejar los datos numéricos en una hoja electrónica, es imprescindible utilizar fórmulas de cálculo que permitan obtener el valor de un elemento en función de otros elementos de la matriz. En definitiva, hay que recurrir a lo que en las Ciencias Matemáticas se denomina *función.*
En breve síntesis, una *función* no es más que una expresión que permite calcular el valor ignorado de una o más variables, a partir de los valores conocidos de otras variables distintas a las iniciales.
Atendiendo al número de variables/incógnita, o variables desconocidas que se evalúan por medio de la expresión, es posible clasificar a las funciones de la siguiente forma:
— *Funciones simples.* Son las que únicamente resuelven el valor de una variable.
— *Funciones de dos valores.* Resuelven el valor de dos variables o incógnitas.
— *Funciones de tres valores.* Resuelven el valor de tres variables o incógnitas.
— ...
En las hojas electrónicas de cálculo se utilizan únicamente funciones simples, ya que tan sólo se pretende obtener el valor de un único elemento.
Otra posible clasificación dentro de las funciones, es la que se obtiene sin más que contar el número de variables conocidas que intervienen en la expresión. En este sentido cabe hablar de:
— *Funciones de una variable.* Son las que sólo utilizan una variable dentro de la expresión de cálculo.
— *Funciones de dos variables.* Utilizan dos variables en la expresión.
— *Funciones de tres variables.* Utilizan tres variables en la expresión.
— ...
En las fórmulas de cálculo que se emplean en las hojas electrónicas, se pueden utilizar funciones de distinto número de variables. Por ejemplo, si en el elemento localizado en la fila 7 y columna 3 de la matriz, se desea obtener el valor resultante de multiplicar el elemento de la fila 7 y columna 1 por la constante 100, y dividir el resultado por el elemento de la fila 7 y columna 2 —ello equivale a calcular en la posición 7,3 el porcentaje que representa la posición 7,1 sobre la posición 7,2—, se utilizará una función simple de dos variables. Su expresión coincidirá con la siguiente:

$$\text{POSICION 7,3} = \frac{\text{POSICION 7,1} \times 100}{\text{POSICION 7,2}}$$



Definición elemental del concepto de función



esta función hace que todas las expresiones referentes al valor se convierten en *"ERROR"*. Esta función sin argumento se invoca sencillamente con:
@ ERROR.

NA

Función análoga a la anterior, con la salvedad de que, en este caso, el valor se convierte en *"NA"* (no disponible). El formato de llamada es:
@ NA.

FUNCIONES MATEMATICAS

Este último grupo integra a las funciones "por excelencia": las especializadas en definir relaciones matemáticas entre los elementos de la hoja electrónica.

Para medir un ángulo en radianes, hay que considerar que la circunferencia total de 360 grados tiene 2 radianes. En consecuencia, cada fracción de 90 grados coincidirá con /2 radianes (=número PI).



ABS

Extrae el valor absoluto de un elemento numérico de la hoja electrónica, prescindiendo de su signo. Ello significa que el resultado coincidirá con la expresión positiva del valor que figure en el argumento de entrada.
@ ABS (elemento numérico).

Una característica de las funciones SIN (seno) y COS (coseno) es que su valor está siempre comprendido entre los valores −1 y 1.

SIN

Función seno. Sirve para calcular el valor del seno del argumento de entrada. Este debe ser un elemento numérico representativo de un ángulo expresado en radianes. Su formato de llamada es:
@ SIN (elemento numérico).

COS

Función coseno. Calcula el coseno trigonométrico del valor de entrada, considerado éste como un ángulo en radianes:
@ COS (elemento numérico).

TAN

Función tangente. Determina la tangente del ángulo cuyo valor en radianes constituye el argumento de entrada:
@ TAN (elemento numérico).

ASIN

Función arcoseno. Calcula el valor en radianes del ángulo cuyo seno coincide con el valor numérico del argumento de entrada. Su formato de llamada es:
@ ASIN (elemento numérico).

ACOS

Función arcocoseno. Determina el valor en radianes del ángulo cuyo coseno coincide con el dato reflejado en el argumento:
@ ACOS (elemento numérico).

ATAN

Función arcotangente. Responde con el ángulo en radianes cuya tangente es igual al valor numérico incluido en el argumento de la función:
@ ATAN (elemento numérico).

LOG 10

Logaritmo decimal. Función de un único argumento que produce como resultado el logaritmo decimal de dicho argumento. El resultado es tal que si se eleva la base 10 al valor producido por la función, se obtendrá el número que aporta el argumento. El formato para invocarla es:
@ LOG 10 (elemento numérico).

LN

Logaritmo neperiano. Calcula el logaritmo neperiano o natural (cuya base es el número "e"=2.718...) del valor expresado en el argumento:
@ LN (elemento numérico).

EXP

Función exponencial. Calcula el valor del número "e" elevado al argumento de entrada; su formato es el siguiente:
@ EXP (elemento numérico).

INT

Parte entera. Esta función se complementa con un único argumento, coincidente con cualquier elemento numérico de la hoja electrónica. Entrega como resultado la parte entera de dicho argumento; para ello elimina del mismo los decimales que pueda incorporar:
@ INT (elemento numérico).

SORT

Raíz cuadrada. El resultado producido por esta función es la raíz cuadrada del valor de su único argumento numérico de entrada. Para invocarla hay que especificar:
@ SORT (elemento numérico).

PI

Esta es una función constante que no requiere argumento; responde dando el valos del número PI: 3,1415926536. Su formulación se reduce a:
@ PI.

